GenoMus: Representing procedural musical structures with an encoded functional grammar optimized for metaprogramming and machine learning

A Preprint

José López-Montes✉
PhD Student E.T.S. Ingeniería Informática
Departamento Ciencias de la
Computación e Inteligencia Artificial
University of Granada
lopezmontes@correo.ugr.es

Miguel Molina-Solana
E.T.S. Ingeniería Informática
Departamento Ciencias de la
Computación e Inteligencia Artificial
University of Granada
miguelmolina@ugr.es

Waldo Fajardo
E.T.S. Ingeniería Informática
Departamento Ciencias de la
Computación e Inteligencia Artificial
University of Granada
aragorn@correo.ugr.es

April 25, 2022

Abstract

We present GenoMus, a new model for artificial musical creativity based on a procedural approach, able to represent and learn compositional techniques behind a musical score. The aim of this model is to build a framework for automatic creativity, easily adaptable to other domains beyond music. The core of GenoMus is a functional grammar designed to cover a wide range of styles, integrating traditional and contemporary composing techniques. In its encoded form, both composing methods and music scores are represented as unidimensional normalized arrays. On the other hand, the decoded form of GenoMus grammar is human-readable, allowing manual edition and the implementation of user-defined processes. Musical procedures (*genotypes*) are defined as functional trees, able to generate musical scores (*phenotypes*). Each subprocess uses the same generic functional structure, no matter what time scale, polyphonic structure, traditional or algorithmic process is being employed. The goal of this highly homogeneous and modular approach is to simplify metaprogramming and to maximize search space. This abstract and compact representation of musical knowledge as pure numeric arrays is designed for a convenient application to different machine learning paradigms.

Keywords: automatic musical composition · metaprogramming · procedural representation of music · artificial creativity · GenoMus

Contents

1 Introduction

> *But there's a big difference between "impossible" and "hard to imagine". The first is about it; the second is about you*!
> —Marvin Minsky [26]

[30] -> interesante para situar la discusion y encuadrar las motivaciones y posibles peligros. Motivations classes: 1. computer programs are written by the composer as an idiosyncratic extension to her own compositional processes; 2. computer programs are written as general tools to aid any composer in the composition of music; 3. theories of a musical style are implemented as computer programs; 4. cognitive theories of the processes

Failures: 1. a failure to specify the precise practical or theoretical aims of research; 2. a failure to adopt an appropriate methodology for achieving the stated aims; 3. a failure to adopt a means of evaluation appropriate for judging the degree to which the aims have been satisfied.

2 Background and related works

> *Lerner appears to believe that transformations that could be carried out by a computer program [...] could not possibly generate anything sensible—and that no program could tell sense from nonsense anyway. The implication [...] is that no computational theory could describe the generation of valuable new ideas, and that only an unanalyzable faculty of "intuition" or "insight" could recognize their value. None of these beliefs is justified.*
>
> —Margaret Boden [5]

hablar de gramaticas generativas

explicar las caracteristicas necesarias para luego enlazar con las caracteristicas que presenta GenoMus

[33] -> Conditions for creativity: Knowledge representation is organised in such a way that the number of possible associations is maximised. A flexible knowledge representation scheme. Similarly Boden (1996) says that representation should allow to explore and transform the conceptual space. Tolerate ambiguity in representations. Allow multiple representations in order to avoid the problem of ?functional fixity?. The usefulness of new combinations should be assessable. New combinations need to be elaboratable to find out their consequences. El articulo de Boden es: What is Creativity. In M. Boden, editor, Dimensions of Creativity, pages 75?118. MIT Press, 1996. (citado antes).

2.1 Composing composers

Metalevel -> [6]

[18] -> interesante la disgresion filosofica sobre la autoria. De Music composition to music recognition. Dice: Is the same? Yo digo: Can computer solutions get us to new styles, to nwe ways or feeling music?s

[22] -> ACTUAL! Algoritmic Music Composition Based on Artificial Intelligence: A Survey

El survey de Vico [32]

[12] -> survey actual de evolutionary music generation (para inicio)

Research in artificial musical intelligence demand for formalized grammars of musical structures. *[introducir cita, y hablar en el background de gramáticas generativas]* Besides, a model of creative mind is required to operate these abstractions. Aesthetic criteria are extremely subjective, furthermore the details of every model of automatic composition impose, consciously or not, a limited search space. Delimiting these boundaries and setting evaluation principles can be seen as metacomposition, namely composing composers.

Composers' interest in musical language pervaded the 20th century aesthetics. Transformation and overcoming of well-established methods inherited from Romanticism led to post-tonal music. Linguistic structuralism applied to musical syntax stimulated relativization and consciousness of compositional procedures. Reversing the logic of this analytic knowledge, the methods of serial dodecaphonic music was the first step for the foundations of an inverse creative strategy: synthesize new styles from the predefinition of new rules.

Computer assisted composition enabled far more complex procedures, tedious or unfeasible to explore by hand. Eventually, composers began to use computers not only for analysis and calculation of complex structures, but for the automation of the creative processes themselves. That fact opened the door to a new approach to composition: a metamusical level characterized by modeling the processes within the minds of composers.

[Reflexiones sobre metacomposicion, el concepto de autoria y consideraciones pedagogicas y humanas de fondo.]

[Interes de la musica en el modelado de creatividad artificial - multidimensionalidad de la percepcion y analisis]

[Sobre la necesidad de usar el metanivel de los procedimientos antes que la partitura]

Many approaches to artificial intelligence applied to the automatic composition of music are modeled using scores as its data source...

[Complejidad del diseno de lenguajes de representacion musical en la composicion asistida por ordenador. Cita de algunas aproximaciones analogas.]

2.2 New challenges for automatic composition

[Repaso rapido de los principales paradigmas de herramientas de CAC para hacer notar como es necesario un metanivel de trabajo]

Citar [27] -> -Pag. 4-5: Buen sumario de los paradigmas -> -5: justifica problema de neural nets con fragmentos largos coherentes

Muy interesante: Open Problems for Genetic Music: [24]

This proposal, beyond the technical details, is a model of augmented creativity from the point of view of the composer. The new paradigm of computation applied to creative tasks is tilting from ordering to the computer "what to do", to saying "what to get", as a starting point to detonate supervised or non-supervised creative processes. So, the goal of GenoMus is to combine a knowledge base of compositional procedures with the maximal freedom of recombination.

Hacer resumen del proyecto completo, referencias a los antecedentes y dejar claro que trata este texto

3 Methods: The GenoMus framework

> *I believe that music today could surpass itself by research into the outside-time category, which has been atrophied and dominated by the temporal category. Moreover this method can unify the expression of fundamental structures of all Asian, African, and European music. It has a considerable advantage: its mechanization—hence tests and models of all sorts can be fed into computers, which will effect great progress in the musical sciences.*
>
> —Iannis Xenakis [38]

3.1 Foundations and requirements

- Requerimientos previos deseados que han determinado el diseño
- Bases del desarrollo (capacidad de interactuar con los resultados musicales, etc.)

Similaridad con [15] como el referente mas cercano. Mostrar diferencias. Tambien cercano a [2] como lenguaje creado por un compositor para autoexpansion.

[9] -> solo por citar una aproximacion hibirda muy actual, pero poco interesante

[10] > aproximacion similar en su planteamiento, pero muy limitada

[19] -> un ejemplo analogo a la generacion automatica de arboles de funciones

Referencia a compositor de referencia -> [38] Ya habla de la posibilidad de expresar la musica como expresiones logical and algebraic a partir de eventos sonicos basicos, a la par que habla de ir "towards a metamusic"

Over the last decades, a plethora of grammars (citarlas) to represent music have been proposed. This framework...

After studying the potential and limitations of previous research (referencia), and based on the experience of several iterations of the core concept, the GenoMus grammar has been designed to satisfy these key features justificar con referencias de donde se extrae estas conclusiones de que sean key features):

Ahora mismo, hay una mezcla de niveles de abstraccion; por ejemplo, 'a', 'g' o 'j' son caracteristicas, mientras que 'd' dice como se debe de hacer.

Mi propuesta simplificada de caracteristicas:

- Representacion modular.

- Representacion compacta (es menor o igual que el resultado que codifica).

- Representacion extensible (se pueden usar subconjuntos de la gramatica)

- Representacion manipulable procedimentalmente.

- Representacion consistente (un codigo siempre genera la misma salida)

- Soporte para autoreferencias internas.

- Aplicabilidad a otros contexto.

a) Grammar based on a symbolic and generative approach to music composition and analysis.

GenoMus is focused on the correspondences between compositional procedures as musical results. It employs the genotype/phenotype metaphor, as many other similar approaches, but in a specific way, discussed below.

b) Style-independent grammar, but able to incorporate complex processes from contemporary techniques.

In any approach to artificial creativity, a representation system is a precondition that restricts the search space and imposes aesthetic biases a priori, either consciously or unconsciously. The design of algorithms to generate music can be ultimately seen as an act of composition itself. With this in mind, our proposal seeks to be as open and generic as possible, to represent virtually any style and enclose any procedure. The purpose of the project is not to imitate styles, but to create results of certain originality, worthy of being qualified as *creative*.

A smooth integration of modern and traditional techniques is one of the purposes of our grammar. GenoMus allows to include any compositional procedure, even those from generative techniques that imply iterative subprocesses, such as recursive formulas, automata, chaos, constraint-based and heuristic searches, L-systems, etc.

c) Optimized modularity for metaprogramming.

Each musical excerpt is generated by a function tree made with a palette of procedures attending all dimensions: events, motifs, rhythmic and harmonic structures, poliphony, global form, etc. All function categories share the same input/output data structure, which ease the implementation of metaprogramming routines encompassing all time scales and polyphonic layers of a composition.

d) Identical encoded representation of both compositional procedures and musical scores as single unidimensional and normalized vectors.

Compared to similar grammars, this is the distinctive feature of GenoMus. Functional expressions and the results of their evaluations are both encoded as sequences of floats within the interval $[0, 1]$. Thus, pairs of complex nested procedures and their corresponding complex polyphonic scores are mapped as flattened unidimensional arrays. This abstract representation of music is suitable to be handled with different machine learning techniques.

e) Alternative decoded format of compositional procedures, as function trees easily understandable and manually editable.

Functional expressions and their encoded format are alternative representations. So, the abstract vectorial format can be converted into a readable and editable text with convenient numeric scales for each parameter.

f) Representation of music able to capture all time scales and polyphonic layers, from expressive details to global form.

The aesthetic potential of computer generated artworks is often obscure, especially when there are aspirations to find original styles and new rules. Human composers usually conceive sequences of notes, agogics, articulation gestures and dynamic expressiveness as a whole. There is no intrinsic value in a sequence of durations and pitches if it lacks of expressive attributes such as articulation and dynamics. Interrelations of different parameters, both in short and long term, are critical to get appealing results.

A good piece of music is much more of than the sum of its parts.[1] Composers often construct a piece by planning interrelations between the details of motifs and the overall structure. This holistic conception of creativity have been obviated in many previous research focused on particular compositional tasks. So, in our framework macro and microformal features are created and transformed as an entire entity from the beginning of the transformational operations.

Preliminar experiments with GenoMus showed that many of the randomly generated musical excerpts exhibited surprising expressive qualities: an interesting dynamic gesture can create a feeling of order and purpose when applied to sequences of notes that otherwise could be assessed as meaningless or too random. So, to discover potential combinations, our framework fuses the generation of completely developed excerpts with the incremental transformation of the details of any element inside them.

g) Support for internal autoreferences.

[1] Beethoven perfectly exemplifies how sublime and huge musical structures can emerge from trivial and seemingly uninteresting musical motifs.

In almost any composition, some essential procedures require the reuse of previously heard patterns. As many pieces consist of transformations and derivations of motifs presented at the very beginning, our framework enables pointing to preceding patterns. At execution time, each subexpression is stored and indexed, being available to be referenced by the following functions of the evaluation chain.

Beyond the benefits of avoiding internal redundancy when there are repeated patterns, the possibility of create internal autorefences of nodes inside a function tree is an indispensable precondition for the inclusion of procedures which demand recursion, reevaluation of subexpressions.

h) Consistency of the correspondences among procedures and musical outputs

To get an increasing knowledge base, correspondences between expressions, encoded representations and resulting music must be always the same, regardless of the subsequent evolution of the grammar and the progressive addition of new procedures by different users.

For encoding musical procedures, each function name maps to a number. But in order to keep the encoded vectors as different as possible, function name indexes are scattered across the interval $[0, 1]$ and registered in a library containing all available functions.

i) Possibility of generating music using subsets of the complete library of compositional procedures.

Before the automatic composition process begins, it can be selected which specific procedures should be included or excluded from it. It can also be defined mandatory functions to be used in all the results proposed by the algorithm.

j) Applicability to other creative disciplines beyond music.

Although this framework is presented for the automatic composition of music, the model can be easily adaptable to other areas where creative solutions are sought. Whenever it is possible to decompose a result into nested procedures, a library of such procedures can be created that takes advantage of their encoding as numerical vectors that serve as input data for machine learning algorithms.

[14] -> Quiza recoger su apunte de las conclusiones: falta el long-term approach y el higher-level concept

Just con [13] -> para la generacion automatica de arboles y gramatica (f mas a fondo)

Una aproximacion tambien muy similar en el sentido de generar metaprogramas -> [34]

Similar pero muy simple -> [1]

too [4] -> para justificar el uso de arboles de funciones, ya que este articulo habla de parsear lenguaje (musical entre otros) para llegar al arbol original.

[17] -> Propone trabajar con 'building blocks' mas grandes, cercano a frases

[Conveniencia del paradigma de programacion funcional para la metaprogramacion. Antecedentes procedimientos compositivos como funciones (referencias a Haskell y LISP en la tradicion)]

3.2 Knowledge representation as one-dimensional arrays

- Vocabulario empleado en la analogía genética
- Detalles sobre los elementos codificados como vectores o como expresiones legibles

Although its target is not only genetic algorithms but a variety of machine learning techniques, our framework uses the evolutionary analogy, similarly as many other automatic composition systems [36, 31, **?**]. The Darwinian metaphor can be confusing, since each system has its particular application of the same terms, sometimes denoting even opposite concepts.

In our functional approach, the key idea is *considering a piece of music as the product of a program* which encloses compositional procedures. Thereupon, the bio-inspired terms in the context of the GenoMus framework are defined as follows:

genotype: Computable tree of compositional procedures.

genotype function: Minimal computable unit of a genotype, designed in a modular way to enable taking other genotype functions as arguments.

decoded genotype: Genotype coded as a readable, understandable and evaluable string.

encoded genotype: Genotype coded as an array of floats $\in [0, 1]$.

phenotype: Music generated by a genotype.

encoded phenotype: Phenotype coded as an array of floats $\in [0, 1]$.

decoded phenotype: Phenotype converted to a format suitable for third-party music software.

The genotype, in its decoded format, is actually a computable expression. Consequently, the automatic writing and manipulation of genotypes can be seen as a metaprogramming process.

The phenotype, as the product of the evaluation of the functional expression content in a genotype, is a sequence of sonic events declared in a format designed to encode music according to a hierarchical structure, made by combining these three formal categories:

event: Single sonic element. An event is a wrapper for a sequence of numeric parameters.

voice: Line (or layer) of music. A voice is a wrapper for a sequence of one or more events, or a wrapper for one or more voices sequentially concatenated, without overlapping.

score: Piece (or passage) of music. A score can be a wrapper for one or more voices, or a wrapper for one or more scores together. Scores can be concatenated sequentially (one after another) or simultaneously (sounding together).

Depending on the final desired output of the generative process, a different parametric structure will be expected for a event. Hence, to allow diverse formats for decoded phenotypes, two more concepts arise:

species: Specific configuration of the internal structure of parameters required to construct an event.

specimen: A genotype/phenotype pair belonging to a species, that produces a piece of music. A specimen data are content in a dictionary along with metadata and additional analytical informations.

Continuing with the biological analogy, a species can be alternatively defined as the group of specimens that share the same parameter structure of their musical events. The species called *piano*, used in the examples, requires four attributes for each event: duration, pitch, articulation and intensity.

A parameter required by a event can be a single value or an array of values (a multiparameter). For instance, in the piano species, an event can contain more than one pitch, to work with chords.

Events build with many parameters can be set. For instance, a species for a very specific electroacoustic setup could need events defined by dozens of features.[2] An event specification can also be extended to other domains beyond music, like visuals, lighting, etc., along with musical events, or standalone.

-> usa el paradigma genotipo fenotipo, aunque de un modo diferente Muy importante: Indirect encoding: IAMUS -> importancia de la evo-devo incremental. El proceso importa, no solo la codificacion del ADN ?Using an effective indirect encoding, a small genotype can potentially specify a large and complex phenotype, accounting for the scalability problem previously mentioned. Additionally, a small change in the genotype can potentially provoke a variety of coordinated changes in the phenotype?

?They are currently being used to a certain extent for automating tasks that demand creativity, proposing different variations to existing solutions, which evolve toward desired design targets, resembling an automated form of brainstorming.?

[35] -> In order to meet this challenge, many researchers are proposing indirect encodings, that is, evolutionary mechanisms where the same genes are used multiple times in the process of building a phenotype.

[Similitud con la programacion funcional: la pieza musical como funcion de funciones.]

3.3 Formal definition of the GenoMus framework

- Visión formal y general de los elementos necesarios para crear el modelo operativo

A GenoMus framework for a species s is defined as the 10-tuple $G_s = \{P, \mathbf{e}, T, i, F, E, L, M, C, \mathbf{d}\}$, where

- P is the set of parameter types necessary to represent the dimensions of an event.

[2] A preliminar test of a species for complex sound synthesis and spatialization where used for the composition of the piece *Microcontrapunctus* [21].

- **e** is the vector of ordered parameter types $\in P$ specifying the format of an event.
- T is the set of genotype function types.
- $i \in T$ is the function type required to init a genotype first node.
- F is the set of all genotype functions registered in a specific library for species s.
- $E \subset F$ is the set of eligible functions selected to build genotypes.
- L is the set of leaf types employed as genotype terminal nodes.
- M is the set of conversion functions mapping specific formats of any parameter type $\in P$ into real numbers $\in [0, 1]$, and their correspondent inverse functions.
- C is the pair of an encoding function to transform genotype functions trees into unidimensional vectors of reals $\in [0, 1]$, and its inverse function.
- **d** is a generic data structure to be taken as input or returned as output by any function $\in F$.

The data content in **d** has the form of a dictionary (actually a JavaScript Object in our implementation), including this properties:

`funcType`**:** Function type according to its output.

`encGen`**:** Encoded genotype.

`decGen`**:** Decoded genotype (the function's own expression as a string).

`encPhen`**:** Encoded phenotype of the generated score.

`phenVoices`**:** Number of voices of the generated score

`phenLength`**:** Number of events of the generated score

(*others*): Additional information, generally related to a self-analysis of musical features that are useful for the next genotype node, such as tempo, rhythm, harmony, etc.

This data structure is what each genotype function expects for every argument, and what is passed to the next one. A crucial item is **decGen**, where a function returns its own code as a string. It allows reevaluations of this code, which is essential to procedures involving iteration, recursion or stochastic processes.

The decoding of GenoMus encoded phenotypes into standard formats for any musical application can be seen as an external operation, not concerning directly this formal definition.

3.4 Musical genotypes, phenotypes and germinal vectors

- Formalización matemática de la forma de los elementos y de cómo se relacionan entre sí

A *germinal vector* determines a decision tree for the core algorithm that leads to an encoded genotype, a valid executable expression that represents musical procedures.

Let D_{gen} be the set of possible decoded genotypes for a given GenoMus species, using the available functions E. Let *dec* be the conversion function that decodes an encoded genotype:

$$dec: \ E_{gen} \rightarrow D_{gen}$$

A germinal vector is a random sequence of n numbers $\in [0, 1]$. Let G_{vec} be the multidimensional search space[3] of all possible of germinal vectors $\mathbf{g} = (g_1, g_2, ..., g_n)$ of any cardinality n, and similarly let E_{gen} be the set of all possible valid encoded genotypes $\mathbf{e_g} = (x_1, x_2, ..., x_m)$:

$$G_{vec} = \{\mathbf{g} \in \mathbb{R}^n \mid 0 \leq g_n \leq 1, n \in \mathbb{N}_{>0}\}$$
$$E_{gen} = \{\mathbf{e_g} \in \mathbb{R}^m \mid 0 \leq x_m \leq 1, m \in \mathbb{N}_{>0}, \exists\, \mathbf{d_g} = dec(\mathbf{e_g})\}$$

[3]Obviously, in a practical implementation, the theoretical infinite germinal vectors are reduced to a finite subset due to computability and memory limits, as well as the number of significant figures used to store and operate on the vectors.

Each germinal vector in G_{vec} is mapped to a valid encoded genotype (that is a valid functional expression which generates an excerpt of music). To create genotypes, the surjective map f converts every germinal vector $\mathbf{g} = (g_1, g_2, ..., g_n) \in G_{vec}$ into an encoded genotype:

$$f : G_{vec} \twoheadrightarrow E_{gen}$$

The application f involves several subprocesses that we will cover next, in such a way that the vector $\mathbf{g}$ generates a deterministic decision tree that leads to the construction of a unique genotype. As the number of choices needed to build a valid encoded genotype E_{gen} rarely match the number of items of a germinal vector G^n_{vec}, truncation and loops are employed:

- If $\mathbf{g}$ has more items than needed, they are ignored, effectively acting as a truncation of the remaining unused part of $\mathbf{g}$.
- If f needs more items than those supplied by $\mathbf{g}$, f reads $\mathbf{g}$ repeatedly from the beginning as a loop, until closing the valid encoded genotype. That implies that even vectors with a single value can be mapped to large functional expressions.

3.5 Encoding and decoding strategies for data normalization

- Rangos de mapeo equuilibrados para los parámetros musicales
- Detalle del uso de golden angle para codificar enteros (Aplicaciones: Espaciado equilibrado de punteros a funciones en el rango 0-1, codificación de enteros en fenotipos (número de voces, de notas, etc.)

3.6 Germinal vectors as retrotranscription of decoded genotypes

- Explicación formal matemática y ejemplos
- Énfasis en la retrotranscripción como un elemento clave del sistema

[11] -> para las resonancias godelianas

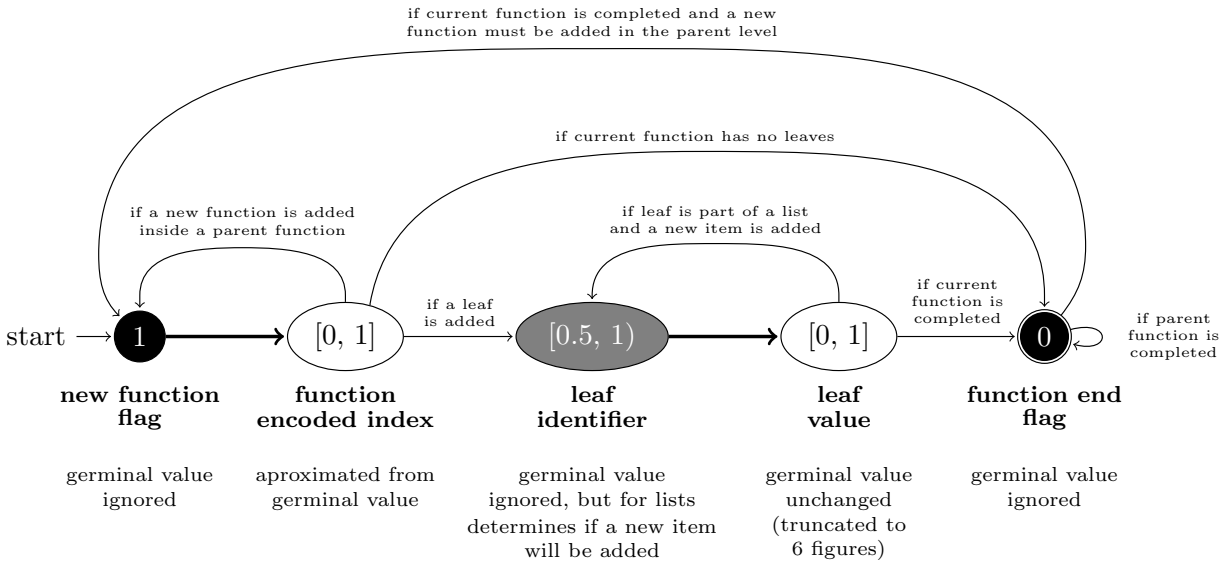

Figure 1: From germinal vector to encoded genotype

3.7 Anatomy of a genotype function

3.8 Function types

These types can either correspond to structural levels of a phenotype, or can attend to specific data substructures needed to feed particular functions.

[Tabla con los tipos de funciones. Figura ilustrando la estructura score/voice/chord]

3.9 Leaves, parameter mapping and readability

3.10 Function libraries

3.11 Specimen data structure

Selected specimens are permanently saved as JSON files in the Specimen catalogue. These JSON files are created according to the data structure showed in Table 1.

3.12 Overview of the model and complementary materials

- Diagrama completo del sistema, incluyendo el proceso de interacción con el usuario
- Enlace a las especificaciiones detalladas y a ejemplos

4 A minimal example: *Clapping Music*

> *[...] in general the rules don't make the music, it is the music that makes the rules. Then, instead of relying only on a set of imperfect rules, why not making use of the source of the rules, that is the compositions themselves?*
>
> —Ramón López de Mántaras [20]

4.1 Procedural representation

To illustrate how GenoMus represents and encodes a piece of music using basic procedures in an abstract format, we will apply the framework to model *Clapping Music*, a famous piece composed by Steve Reich in 1972. As many of the pieces of the composer, it begins with a minimal motif, which undergoes simple transformations that have big consequences in the overall form. Two performers start the piece repeating a clapped pattern. One of the performers removes the first note of the pattern after each cycle of 8 repetitions[4], creating a phase shift among both voices. The work finished when both lines are in phase again.

The piece was modeled using the *piano* species which, according to the formalism stated in section 3.3, is defined as the set $G_{piano} = \{P, \mathbf{e}, T, i, F_{piano}, E, L, M, C, \mathbf{d}\}$, where

- $P = \{notevalue, midipitch, articulation, intensity, quantized, ...\}$ [parameter types]
- $\mathbf{e} = (notevalue, midipitch, articulation, intensity)$ [event format]
- $T = \{scoreF, voiceF, quantizedF, lnotevalueF, lmidipitchF, ...\}$ [genotype function types]
- $i = scoreF$ [genotype function type for first node]
- F_{piano} is the set of all genotype functions registered in the specific library for the *piano* spieces.
- $E = \{s2V, vSlice, vRepeatV, vConcatV, cMotifLoop, vAutoref, ln, lm, la, li, q\}$ [eligible functions]
- $L = \{lnotevalueLeaf, lmidipitchLeaf, larticulationLeaf, lintensityLeaf, quantizedLeaf\}$ [leaves]
- $M = \{norm2notevalue, notevalue2norm, norm2midipitch, ...\}$ [conversion functions]
- $C = \{encodeGenotype, decodeGenotype\}$ [encoding/decoding functions]
- $\mathbf{d}$ is the data structure to serve as inputs and outputs for all functions $\in F_{piano}$.

[4] According to the score, each loop can consist of 8 or 12 repetitions. We have chosen 8 repetitions for the sake of clarity and because is the preferred version for most of recordings.

`metadata`	Data to identify and classify the specimen within the catalogue. `specimenID`, `permalink`, `GenoMusVersion`, `creationTimecode`, `user`, `globalRating`, `iterations`, `millisecondsElapsed`, `encGenotypeLength`, `encPhenotypeLength`, `decPhenotypeDuration`, `depth`, `totalFunctions`
`initialConditions`	Set of initial conditions to be satisfied by the generative process. `species`, `functionLibrary`, `eligibleFunctions`, `requestedProfile`, `genMaxLength`, `phenMinLength`, `phenMaxLength`, `maxIterations`
`encodedGenotype`	Array of floats $\in [0, 1]$.
`decodedGenotype`	String with the genotype function tree.
`formattedGenotype`	String with the decoded genotype formatted with tabulations and line breaks to display the function tree in a readable format.
`leaves`	Leaf values and their corresponding position in the encoded genotype.
`subexpressions`	Enumerated list of all functional substructures within a genotype, classified by output type.
`encodedPhenotype`	Array of floats $\in [0, 1]$.
`decodedPhenotype`	Phenotype converted to the target third-party formats.
`autoanalysis`	Music features automatically analyzed by the genotype functions. `autoreferenceDensity`, `variability`, `rhythmDensity`, `rhythmComplexity`, `polyphony`, `chromaticism`, `dissonance`, `disjunctivity`, `modalChroma`, `tonalStability`, `tessituraDispersion`, `globalArticulation`, `articulHomogeneity`, `globalDynamics`, `dynamicHomogeneity`, ...
`evolutionLog`	Log covering the history of actions and manipulations operated on the specimen.
`humanEvaluation`	Human evaluation of different musical, aesthetic and emotional aspects of the specimen. This item stores individual assessments of different users, and give an average rating for different musical aspects `aestheticValue`, `originality`, `mood`, `emotionalIntensity`, `individualEvaluations`

Table 1: Specimen data structure

Based on this architecture, there exist several ways to model the piece, since its patterns can be obtained by different methods.[5] The Figure 2 shows the score and one of the possible analytical deconstructions in patterns.

[5]Comparing different ways of generating this work is an interesting question beyond this paper's scope. Each model can correspond to alternative manners of perceiving structural aspects. A different analysis and procedural model of *Clapping Music* can be found in [16]. To create our model, we looked for the concisest code assembling simple and generic operations.

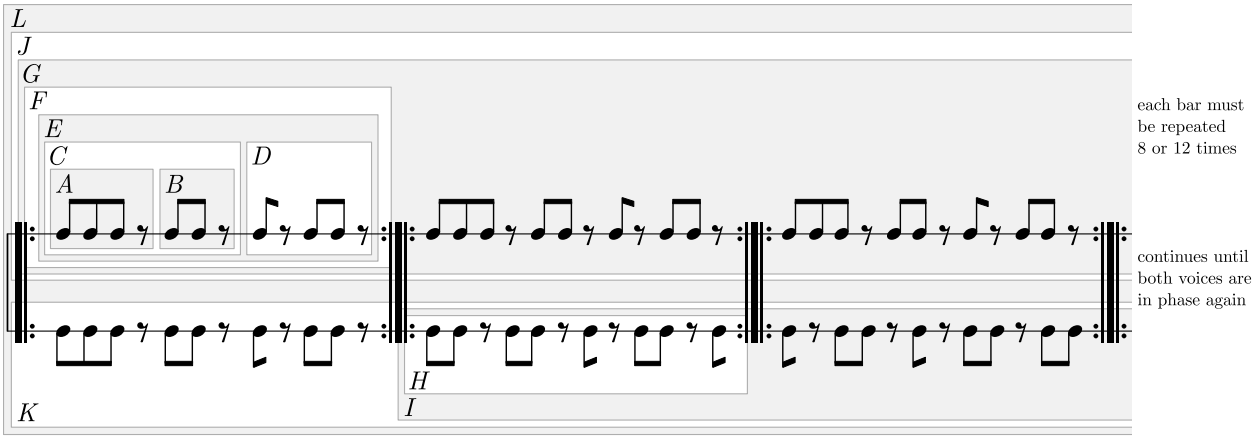

Figure 2: Structures of S. Reich's *Clapping Music*.

Our model derivates the whole composition from transformations of the first four notes (motif A), appling five generic operations using the genotype functions explained in Table 2. The remaining functions are mere containers for different parameter types.

function name	arguments type	output type	description
`vMotifLoop`	(*lnotevalueF*, *lmidipitchF*, *larticulationF*, *lintensityF*)	*voiceF*	Creates a sequence of events based on repeating lists. The number of events is determined by the longest list. Shorter lists are treated as loops.
`vConcatV`	(*voiceF*, *voiceF*)	*voiceF*	Concatenates two voices sequentially.
`vRepeatV`	(*voiceF*, *quantizedF*)	*voiceF*	Repeats a voice a number of times.
`vSlice`	(*voiceF*, *quantizedF*)	*voiceF*	Removes a number of events at the beginning or at the end of a voice.
`s2V`	(*voiceF*, *voiceF*)	*scoreF*	Joins two voices simultaneously.

Table 2: Genotype functions used to model *Clapping Music*.

Our framework has been implemented by using JavaScript for the core code, and Max[6] with its bach[7] package to interact in realtime with the results of the generative processes. The complete work was recreated by the decoded genotype displayed in Listing 1. The uppercase letters in the comments refer to the patterns analyzed in Figure 2.

```
s2V(                            // score L: joins the 2 voices vertically
  vSlice(                       // voice J: slices last cycle due to phase shift
    vRepeatV(                   // phase G: F 13 times
      vRepeatV(                 // cycle F: E 8 times
        vConcatV(               // pattern E: C + D
          vConcatV(             // motif C: A + B
            vMotifLoop(         // core motif A: 3 8th-notes and a silence
              ln(1/8),          // note values
              lm(65),           // pitch (irrelevant for this piece)
```

[6] https://www.cycling74.com
[7] https://www.bachproject.net

```
10              la(50),           // articulation
11              li(60,60,90,0)), // intensities (last note louder for clarity)
12            vSlice(             // motif B: A with 1st note sliced
13              vAutoref(0),
14              q(1))),
15          vSlice(               // motif D: C with first two notes sliced
16            vAutoref(3),
17            q(2))),
18        q(8)),
19      q(13)),
20    q(-12)),
21  vConcatV(                     // voice K: F + H
22    vAutoref(7),
23    vRepeatV(                   // phase I: H 12 times
24      vSlice(                   // cycle H: F without 1st note, for phase shift
25        vAutoref(10),
26        q(1)),
27      q(12))))
```

Listing 1: Decoded genotype of *Clapping Music* model.

The arguments of the `vAutoref` functions refers to a list of subexpressions stored and updated after the evaluation of each genotype function. An autoreference can refers only to the available subexpressions indexed at the moment of its evaluation. That implies that in the constructive process a function will only reuse previous material of the musical timeline. This reflects how works the perception and memory of music, establishing interrelations with preceding elements.

To clarify the autoreferences inside the genotype, Table 3 enumerate all the subexpressions stored at the end of the genotype evaluation. Only the subexpressions for *voiceF* function type are showed, although there exist for all categories. Autoreferences can refer to foregoing autoreferences, as ocurred at line 25, where `vAutoref(10)` points to `vAutoref(7)`, which in turn points to a subexpression that contains two more autoreferences.

0. `"vMotifLoop(ln(0.125),lm(65),la(49),li(60,60,90,0))"`
1. `"vAutoref(0)"`
2. `"vSlice(vAutoref(0),q(1))"`
3. `"vConcatV(vMotifLoop(ln(0.125),lm(65),la(49),li(60,60,90,0)),vSlice(vAutoref(0),q(1)))"`
4. `"vAutoref(3)"`
5. `"vSlice(vAutoref(3),q(2))"`
6. `"vConcatV(vConcatV(vMotifLoop(ln(0.125),lm(65),la(49),li(60,60,90,0)),vSlice(vAutoref(0),q(1))),vSlice(vAutoref(3),q(2)))"`
7. `"vRepeatV(vConcatV(vConcatV(vMotifLoop(ln(0.125),lm(65),la(49),li(60,60,90,0)),vSlice(vAutoref(0),q(1))),vSlice(vAutoref(3),q(2))),q(8))"`
8. `"vRepeatV(vRepeatV(vConcatV(vConcatV(vMotifLoop(ln(0.125),lm(65),la(49),li(60,60,90,0)),vSlice(vAutoref(0),q(1))),vSlice(vAutoref(3),q(2))),q(8)),q(13))"`
9. `"vSlice(vRepeatV(vRepeatV(vConcatV(vConcatV(vMotifLoop(ln(0.125),lm(65),la(49),li(60,60,90,0)),vSlice(vAutoref(0),q(1))),vSlice(vAutoref(3),q(2))),q(8)),q(13)),q(-12))"`
10. `"vAutoref(7)"`
11. `"vAutoref(10)"`
12. `"vSlice(vAutoref(10),q(1))"`
13. `"vRepeatV(vSlice(vAutoref(10),q(1)),q(12))"`
14. `"vConcatV(vAutoref(7),vRepeatV(vSlice(vAutoref(10),q(1)),q(12)))"`

Table 3: Subexpressions of *voiceF* function type stored during the evaluation of *Clapping Music* genotype.

The Figure 3 displays the functional tree of this decoded genotype, along with its internal autoreferences. The tree is represented in inverse order, from left to right, to reflect how substructures are feeding to subsequent functions which construct larger musical patterns, as well as the subexpressions indexing order.

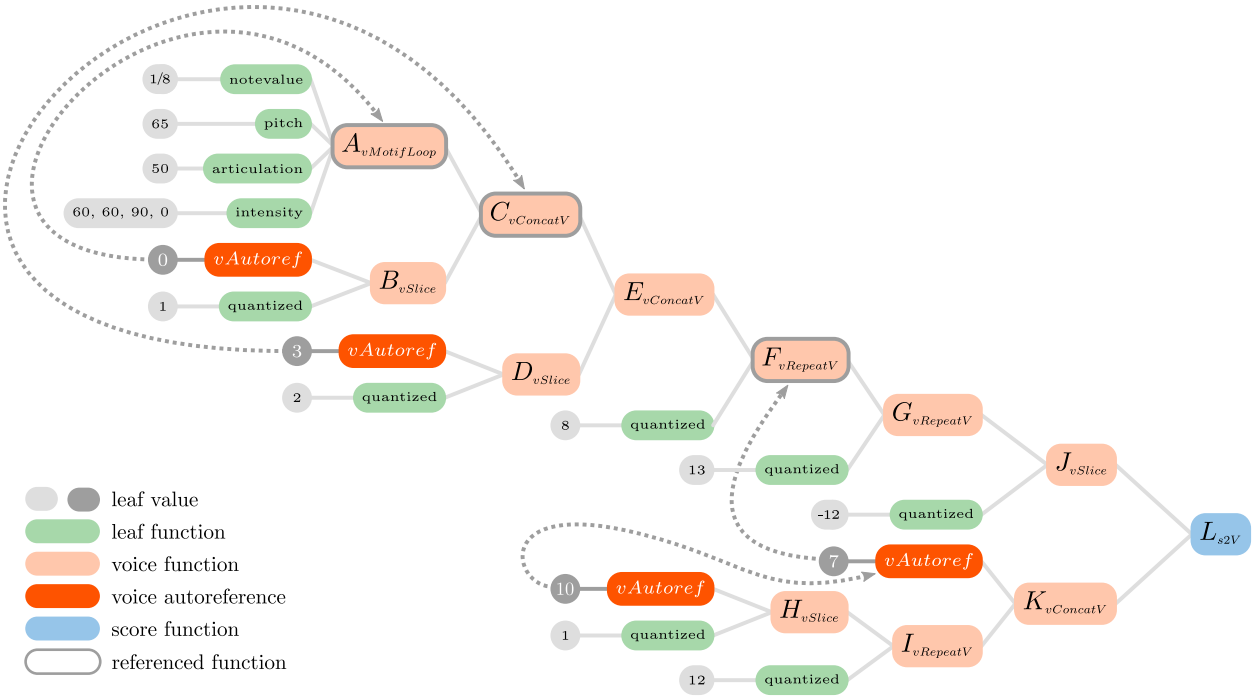

Figure 3: Functional tree of *Clapping Music* genotype.

The encoded genotype is an unidimensional numeric vector. To get a better insight into the correspondences between procedures and music results, we visualize arrays of floats as rows of segments: lengths correspond to values within interval $[0, 1]$; colors are mapped to distinguish slight differences among very similar values, so the patterns have a distinctive visual identity. The encoded phenotype of our *Clapping Music* model consist of 117 values. Figure 4 shows the visualization of this abstract representation of the composition.

Figure 4: Encoded genotype visualization of *Clapping Music* model.

4.2 Extraction of new functions from music

The genotype of a whole composition like this (or only part of it) can be automatically flattened to create a new genotype function where all leaves are assembled as a single array of arguments[8]. For instance, a new function called `sClapping` could be created to be handled as a new procedure, abstracted from the original piece:

[8]The numeric values that feed arguments for the autoreferences are excluded from the arguments array of the new derivate function because of their structural character. These number must be immutable to preserve the internal consistency of the procedure.

function name	arguments type	output type	description
`sClapping`	(*lnotevaluesF*, *lmidipitchF*, *larticulationF*, *lintensityF*, *quantizedF*, *quantizedF*, *quantizedF*, *quantizedF*, *quantizedF*, *quantizedF*, *quantizedF*)	*scoreF*	Creates two voices with a repeated pattern with a progressive phase shift of the second voice created by slicing notes at the beginning of the pattern.

5 Evaluation and evolution

> *Edward Fredkin suggested to me the theory that listening to music might exercise some innate map-making mechanism in the brain. When I mentioned the puzzle of music's repetitiousness, he compared it to the way rodents explore new places: first they go one way a little, then back to home. They do it again a few times, then go a little farther. They try small digressions, but frequently return to base. Both people and mice explore new territories that way, making mental maps lest they get lost. Music might portray this building process, or even exercise those very parts of the mind.*
>
> —Marvin Minsky [25]

[3] para el problema del ?fitness bottleneck? con la eval. humana

[8] -> IMPORTANTE distincion entre Genetic Programming and Gen. Algorithms: 'An overview of earlier studies in EC for musical composition is offered in [12], determining that Ge- netic Programming (GP) methods perform better than those that use Genetic Algorithms (GA). This may be unsurprising as GP methods use a tree-based structure whereas GAs are limited to a linear string in their representation. Hence, GP can represent more complex representations and operations | some- thing that would be very useful in representing music.'

[23] -> The evolution of a population offers so much scope and possibility that it is reminiscent of the music creation process a solution is not linearly determined but instead emerges from a uid, incremental process.

[26] - > interesantes perspectivas de la evaluacion, relacionadas con los prejuicios apuntados por Minsky

Citar [7] para -> Ejemplo de sistema basado en programacion textual, para introduccion manual de expresiones complejas -> A menudo son sistemas creados por los propios compositores como medio de extender su estilo

Buscar donde meter el asunto de la hibridacion Otro ejemplo de hibrido -> [9]

El texto de [28] segun Mantaras, usan fitness basado en contorno, intervalica, y otras cuestiones cuantificables

When modeling artistic creativity with algorithms, probably the most evasive issue to address is programming fitness functions. By definition, the assessment of a piece of art can only be made from a subjective point of view, since the goal of art is to provoke inner and personal reactions. These individual responses are very dependent on cultural and social context. However, provided enough data some predictions can be made about the expected rating for a new piece.

In GenoMus, we divide evaluation of each specimen in two categories:

- Autoanalytic profile: objective analysis of a set of musical features, such as variability, rhythmic complelxity, tonal stability, global dissonance index, level of inner autoreference, etc.
- Human evaluations: subjective ratings made by human users, attending to aesthetic value, originality, mood and emotional intensity. This informations are stored individually and together as global statistics.

The self-analysis contained in every specimen allows to measure distance and similaritiy to other specimens, as well as to classify results and to drive evolution processes.

Defining how to evaluate and select results is now the most creative effort, and can be identified with the act of composition itself, since composing music is ultimately making choices.

- Esta propuesta de gramatica posibilita la implementacion y competencia de diferentes sistemas de evaluacion
- Design of evaluation methods as the crucial act of composition
- Objective vs. subjective evaluation
- What to learn?
- Evolutionary paradigm as the most promising

5.1 Evolutionary paradigm

Determining how to evolve and mutate an specimen towards a best version is crucial question too. Starting from a simple motif, endless evolution paths can lead to satisfying results based on heuristic approaches, using accumulated knowledge from examples, human ratings and automatic self-analysis.

The GenoMus grammar is designed to favor the broadest diversity of combinations and transformations. Genetic algorithms are suitable for the automation of an incremental exploration and selection of multiple ways. A GenoMus decoded genotype tree expression can be trasnformed using these methods:

- createGen
- mutateLeaves
- growTrunk
- growBranch
- insertBranch
- flattenBranch
- pruneBranch
- splitGen

Approaches based on neural networks need a very controlled format of data and big training datasets. The encoded genotype format of GenoMus can represent any piece of music as a simple unidimensional sequence of normalized floats, which can be profitable for techniques as recurrent neural networks (ref. to LMSTD), able to learn patterns from sequential streams of data.

5.2 Scalability

- Como conjugar universalidad de las expresiones con optimizacion para tener los vectores codificados con mayores diferencias entre si.
- Estrategias de caracterizacion de perfiles estilisticos
- El problema del mapeo de funciones y su extensibilidad
- Como establecer una base de datos de conocimiento
- Metricas automatizadas de ciertos resultados

5.3 Safety through encoding

5.4 Integrating traditional and contemporary techniques

5.5 Expanding our musical perception

[37] -> para afirmar que GenoMus reune diferentes paradigmas en sus arboles multicapa. Estos paradigmas segun el articulo son: Analytic, Transformational and Generative

[A genotype como un arbol multiagente, que puede incorporar nodos con funciones de todo tipo: analiticos, recursivos, de constraints, etc. Una vez se tiene un marco de funcion, todo puede caber en el arbol de procesos.]

Justificar con [20]: -> 2: En los 50 era logico excliur la parte expresiva. Los modelos Markovianos son de resultados muy pobres. Ahora debe estar incluida? IMPORTANTE: -> referencia a Minsky, que plantea la posibilidad de los multiagentes, que puedan actuar sobre bloques mayores de musica, y que sean a veces solo analiticos -> ?the rules dont make the music, it is the music that makes the rules? -> Truly creative! para el final

6 Conclusions and ongoing work

> *A generation later, we should be experimenting on programs that write better programs to replace themselves. Then at last it will be clear how foolish was our first idea—that never, by their nature, could machines create new things.*
> —Marvin Minsky [26]

[29] -> concluye que los avances prometedores vendran de la integracion de sistemas diferentes

De la referencia citada al principio [22]: ?Symbolic AI methods still have not enough rules therefore, they hardcoded in limited database knowledge. This could be extended with machine learning techniques but the lack of an automatic evaluation method makes it difficult to solve.?

The artistic results of every algorithm designed for automated composition are strongly constrained by their own representation system of musical data. This paper presents GenoMus, a framework for the exploration of artificial musical creativity based on a generative grammar focused on the abstraction of creative processes as a metalevel of compositional tasks. We define musical genotypes as functional nested expressions, and phenotypes as the pieces created by evaluating these computable expressions. GenoMus' grammar is designed to ease the combination of fundamental procedures behind very different styles, ranging from basic to complex contemporary techniques, particularly those able to produce rich output from very simple recursive algorithms. At the same time, maximal modularity is provided to simplify metaprogramming routines to generate, assess, transform and categorize the selected musical excerpts. The system is conceived to maintain a long term interrelation with different users achieving individual musical styles. This proposed grammar can also be an analytic tool, from the point of view of composition as computation, considering that the best analysis of a piece is the shortest precise description.

Cuestiones interesantes:

- ?Cuantas funciones primitivas son necesarias para generar musica en un determinado estilo? Hay innumerables expresiones funcionales diferentes que pueden generar la misma musica. Se puede deducir que la expresion funcional mas breve es el mejor analisis. Se pueden ver diferentes paradigmas de ensenanza/aprendizaje de la musica con estos modelos.
- ?Como puede hacerse ingenieria inversa automatizada para extraer estructuras desde la musica?

References

[1] ANDO, D., DAHLSTED, P., NORDAHL, M. G., AND IBA, H. Interactive GP with tree representation of classical music pieces. In *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, pp. 577–584.

[2] ARIZA, C. *An Open Design for Computer-Aided Algorithmic Music Composition: athenaCL*. Dissertation.Com, 2005.

[3] BILES, J. GenJam: A genetic algorithm for generating jazz solos. In *Proceedings of the 1994 International Computer Music Conference, IGMA, San Francisco* (1994), pp. 131–137.

[4] BOD, R. The data-oriented parsing approach: Theory and application. In *Computational Intelligence: A Compendium*, J. F. J. Fulcher and L. C. Jain, Eds. Springer-Verlag Berlin Heidelberg, 2008, pp. 330–342.

[5] BODEN, M. A. *Dimensions of Creativity*. The MIT Press, 1996, ch. What Is Creativity?, pp. 75–118.

[6] BUCHANAN, B. G. Creativity at the Metalevel (AAAI-2000 Presidential Address). *AI Magazine 22*, 3 (2001), 13–28.

[7] BURTON, A. R. *A Hybrid Neuro-Genetic Pattern Evolution System Applied to Musical Composition*. PhD thesis, University of Surrey, 1998.

[8] BURTON, A. R., AND VLADIMIROVA, T. Generation of musical sequences with genetic techniques. *Computer Music Journal 23*, 4 (Dec. 1999), 59–73.

[9] CRAWFORD, R. *Algorithmic Music Composition: A Hybrid Approach*. Northern Kentucky University, 2015.

[10] DE LA PUENTE, A. O., ALFONSO, R. S., AND MORENO, M. A. Automatic composition of music by means of grammatical evolution. In *Proceedings of the 2002 conference on APL array processing languages: lore, problems, and applications - APL '02* (2002), ACM Press.

[11] DE LEMOS ALMADA, C. Gödel-vector and gödel-address as tools for genealogical determination of genetically-produced musical variants. In *Computational Music Science*. Springer International Publishing, 2017, pp. 9–16.

[12] DOSTÁL, M. Evolutionary music composition. In *Handbook of Optimization*. Springer Berlin Heidelberg, 2013, pp. 935–964.

[13] DREWES, F., AND HÖGBERG, J. An algebra for tree-based music generation. In *Proc. 2nd Intl. Conf. on Algebraic Informatics, Lecture Notes in Computer Science. This issue* (2007).

[14] HERREMANS, D., CHUAN, C.-H., AND CHEW, E. A functional taxonomy of music generation systems. *ACM Comput. Surv. 50*, 5 (Sept. 2017), 69:1–69:30.

[15] HOFMANN, D. M. A genetic programming approach to generating musical compositions. In *Evolutionary and Biologically Inspired Music, Sound, Art and Design*. Springer International Publishing, 2015, pp. 89–100.

[16] HOFMANN, D. M. *Music Processing Suite Documentation*, version 1.5.1 ed., September 2019.

[17] JACOB, B. L. Composing with genetic algorithms. In *Proceedings of the 1995 International Computer Music Conference, ICMC 1995, Banff, AB, Canada, September 3-7, 1995* (1995).

[18] JACOB, B. L. Algorithmic composition as a model of creativity. *Organised Sound 1*, 3 (Dec. 1996), 157–165.

[19] LAINE, P., AND KUUSKANKARE, M. Genetic algorithms in musical style oriented generation. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence* (1994), IEEE.

[20] LOPEZ DE MANTARAS, R. Making music with ai: Some examples. In *Proceedings of the 2006 Conference on Rob Milne: A Tribute to a Pioneering AI Scientist, Entrepreneur and Mountaineer* (Amsterdam, The Netherlands, The Netherlands, 2006), IOS Press, pp. 90–100.

[21] LÓPEZ-MONTES, J. Microcontrapunctus: metaprogramación con GenoMus aplicada a la síntesis de sonido. *Espacio Sonoro*, 48 (May 2016).

[22] LOPEZ-RINCON, O., STAROSTENKO, O., AND MARTIN, G. A.-S. Algoritmic music composition based on artificial intelligence: A survey. In *2018 International Conference on Electronics, Communications and Computers* (2018), IEEE.

[23] LOUGHRAN, R., AND O'NEILL, M. Generative music evaluation: Why do we limit to 'human'?

[24] MCCORMACK, J. Open problems in evolutionary music and art. In *Applications of Evolutionary Computing, EvoWorkshops 2005: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC, Lausanne, Switzerland, March 30 - April 1, 2005, Proceedings* (2005), pp. 428–436.

[25] MINSKY, M. Music, mind, and meaning. *Computer Music Journal 5*, 3 (1981), 28.

[26] MINSKY, M. Why people think computers can't. *AI Magazine 3*, 4 (1982), 3–15.

[27] NIERHAUS, G. *Algorithmic Composition: Paradigms of Automated Music Generation*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[28] PAPADOPOULOS, G., AND WIGGINS, G. A genetic algorithm for the generation of jazz melodies. In *Proceedings of STeP 98* (1998), pp. 7–9.

[29] PAPADOPOULOS, G., AND WIGGINS, G. Ai methods for algorithmic composition: A survey, a critical view and future prospects. In *AISB Symposium on Musical Creativity* (1999), pp. 110–117.

[30] PEARCE, M., MEREDITH, D., AND WIGGINS, G. Motivations and methodologies for automation of the compositional process. *Musicae Scientiae 6*, 2 (Sept. 2002), 119–147.

[31] Quintana, C. S., Arcas, F. M., Molina, D. A., Rodriguez, J. D. F., and Vico, F. J. Melomics: A case-study of AI in spain. *AI Magazine 34*, 3 (Sept. 2013), 99.

[32] Rodriguez, J. D. F., and Vico, F. J. AI methods in algorithmic composition: A comprehensive survey. *CoRR abs/1402.0585* (2014).

[33] Rowe, J., and Partridge, D. Creativity: a survey of AI approaches. *Artif. Intell. Rev. 7*, 1 (1993), 43–70.

[34] Spector, L., and Alpern, A. Induction and recapitulation of deep musical structure. In *In Proceedings of the IJCAI-95 Workshop on Artificial Intelligence and Music*, pp. 41–48.

[35] Stanley, K. O., and Miikkulainen, R. A taxonomy for artificial embryogeny. *Artificial Life 9*, 2 (2003), 93–130.

[36] Sulyok, C., Harte, C., and Bodó, Z. On the impact of domain-specific knowledge in evolutionary music composition. In *Proceedings of the Genetic and Evolutionary Computation Conference on GECCO'19* (2019), ACM Press.

[37] Wooller, R., Brown, A. R., Miranda, E., Diederich, J., and Berry, R. A framework for comparison of process in algorithmic music systems. In *Generative Arts Practice* (Sydney, Australia, 2005), B. David and E. Ernest, Eds., Creativity and Cognition Studios, pp. 109–124.

[38] Xenakis, I. *Formalized Music: Thought and Mathematics in Composition*. Indiana University Press, 1971.