Free Jazz in the Land of Algebraic Improvisation

Chiriţă, Claudia Elena; Fiadeiro, Jose

Published in:
Proceedings of the Seventh International Conference on Computational Creativity, UPMC, Paris, France, June 27 - July 1, 2016.

Publication date:
2016

Free Jazz in the Land of Algebraic Improvisation

Claudia Elena Chiriţă, José Luiz Fiadeiro
Dept. of Computer Science, Royal Holloway University of London, UK
claudia.elena.chirita@gmail.com, jose.fiadeiro@rhul.ac.uk

Abstract

We discuss the connection between free-jazz music and service-oriented computing, and advance a method for formal, algebraic analysis of improvised performances; we aim for a better understanding of both the creative process of music improvising and the complexity of service-oriented systems. We formalize free-jazz performances as complex dynamic systems of services, building on the idea that an improvisation can be seen as a collection of music phase spaces that organise themselves through concept blending, and emerge as the performed music. We first define music phase spaces as specifications written over a class of logics that satisfy a set of requirements that make them suitable for dealing with improvisations. Based on these specifications we then formalize free-jazz performances as service applications that evolve by requiring other music fragments to be added as service modules to the improvisation. Finally, we present a logic for specifying free jazz based on one of Anthony Braxton's graphic notations for composition notes.

Introduction

Complex dynamical systems, more than being simply complicated, are systems whose structure is intrinsically unpredictable, as they are open to redesign. Their emergent behaviour is dictated by the interconnections and the interactions of cooperation and competition between their entities. These systems are based on non-mereological composition, and thus should be seen as more than the sum of their parts. We argue that both free-jazz performances and service-oriented systems exhibit the characteristics enumerated above and we thus adhere to the tenet expressed in works such as (Borgo and Goguen, 2005), (Borgo, 2005), and (Blackwell and Young, 2004), stating that free-jazz performances are instances of complex systems.

What is free jazz? As it happens more often than not with complex systems and creative processes, free jazz usually gets its diagnosis *per exclusionem*, as it is easier to be understood as the sum of things it is not, rather than of the things it is. We adopt the stance of Mazzola and Cherlin (2008), who propose a positive characterisation of free jazz, instead of the usual negative definition: free jazz is the form of jazz in which the performers are the only ones held accountable for the music that is being played, since (generally) no (standard) notations are followed. The music results from a dynamic, complex game that changes its rules throughout the performance. The success of the game is determined by the identity that emerges from both coherence and conflict – the emergent "dynamical orderings" of the music "that are both surprising and comprehensible" (Borgo and Goguen, 2005). As highlighted in (Borgo, 2005), free jazz is by no means random or lacking rules, even if the evolution of an improvising act is a priori unpredictable due to its transforming constraints and rules: the standards of quality are high, albeit different from the ones of traditional music. Free-jazz improvising is not typically pursuing the classical rhythms, harmonies, or melodies; its valuable aspects are rather the pervading creativity, the discovery of new musical dimensions, the emergence of a collective purpose, or the unexpected synchronizations that interrupt divergence moments. These features seem to challenge (or disregard at least) the existing means of analysing and evaluating conventionally notated music. This is why we believe new tools are required for studying the phenomenon of self-organising music.

Computational models for free jazz Even if we do not intend to address to the utmost the complexity of musical improvisations, we focus on the dynamics of these performances by means of formal, algebraic methods for complex systems. What differentiates our paper from other previous work is the way we approach complexity. We formalize free-jazz performances as complex dynamic systems of services, building on the idea that an improvisation can be seen as a collection of music phase spaces that organise themselves through concept blending, and emerge as the performed music. The music improvised up to a point plays the role of a service application, while all the music fragments that could continue the performance are seen as external service providers; these offer their intrinsic characteristics as services meant to satisfy the 'needs' of the ongoing music act. In (Borgo and Goguen, 2005) free-jazz performances are modelled as non-linear dynamical systems of equations, and in (Blackwell and Young, 2004) as swarm optimization processes. It is worth mentioning that even if optimality is hardly a global feature of free jazz, as improvisation is almost never concerned with meeting objectives in an optimal way, all these computational approaches exploit optimization techniques to model a step in the evolution of the system. Borgo (2005) expresses the "sink or swim" character of improvisations as the "sync or swarm" behaviour of a complex system: either we select the states with the best fitness,

or we decide to let a process end. The swarm formalization aims to echo the emergence of a collective direction of the music performance as a whole, despite the seemingly divergence of its individual components. Although we choose to depart from the randomness inherent to swarm optimization, our approach could be aligned to the *Live Algorithms for Music* manifesto of Blackwell and Young (2004) if, in the modular view of these complex systems, we replaced the "Swarmer" component with the service-oriented framework that we propose. Another notable difference between this swarm approach and the model we advance is the fact that the object of our optimisation is intrinsic to the music: since the constraints are not extraneous, no input (to which the music that we are creating should be compared) is needed.

The study of Ramalho and Ganascia (1994), proof of the long-standing interest in modelling and simulating the musical creativity of improvisations, states similar claims to our assumptions on knowledge and reasoning in jazz performances, such as the fact that musical actions depend on contexts that evolve over time, and that musicians integrate rules and constraints into their actions dynamically. Moreover, it sets similar goals with respect to simulating creativity: obtaining a suitable trade-off between the 'flexibility and randomness' and the 'control and clear semantics' in modelling creativity in terms of classical problem solving.

Modelling aims The primary goal of the model that we propose for free-jazz improvisations falls, according to the taxonomy of motivations for the formalization and automation of music compositional processes defined by Pearce, Meredith, and Wiggins (2002), in the domain of computational modelling of music cognition and musical creativity – see (Wiggins et al., 2009). This includes studies having both cognitive motivations and musicological goals that are not focused on generating aesthetically appealing music or obtaining useful compositional tools, but are rather interested in the degree in which a model serves the comprehension of the cognitive processes within composition and improvisation. Although our main aim is not to propose and evaluate hypotheses on stylistic properties of jazz compositions, on a secondary level, our study lies at the intersection of computational modelling of musical styles and design of compositional tools: the framework could be implemented and thus used to create new computationally creative music systems.

Free-jazz semiotics: music phase spaces We follow the lines of (Borgo and Goguen, 2005) and regard improvisation processes as self-organisational systems of musical *phase spaces*. We consider that the continuous flow of a free improvisation can be segmented into musical sections (regions of a phase space) that capture a distinct musical feature or that have a certain level of cohesion – a prominent qualitative character (related to the rhythm, tempo, timbre etc.). The passage between these sections plays the important role of a bifurcation in the evolution of the modelled improvisation, and will be referred to as a *phase transition*.

The phase space of a system is understood in (Borgo and Goguen, 2005) as a multi-dimensional map which facilitates the description of the dynamics of the given system. The number of dimensions is given by the number of musical variables. The standard music notation captures, for example, a small number of dimensions: time, pitches, and other marks regarding the tempo or other details. One might think that since free jazz does not commit to notations, and since it claims to be more flexible about tonalities and timbre, we would have to deal with phase spaces having large dimensions. What Borgo and Goguen (2005) propose is actually a reduction of the unnecessary variables. We abstract over this representation of the phase spaces, and consider them simply as "musical idea spaces", or semiotic spaces. We loosen the algebraic formalisation of semiotic spaces of Goguen (1999) by considering that a musical phase space could be described in principle through the use of algebraic specifications such as sets of sentences over a given logic – see (Sannella and Tarlecki, 2012). Thus, one should think of a music phase space as a collection of music fragments that share certain salient features and could be played at a certain moment in the evolution of the improvisation. Examples of phase spaces, together with a formal definition of the concept, are discussed in the next section.

Service binding as concept blending The notion of concept blending as used by Goguen (1999) plays a key role in defining the composition of musical phase spaces, which in turn determines the outcome of the improvisation process. Similarly to the studies of Eppe et al. (2015) and Kaliakatsos-Papakostas et al. (2014) on the role of conceptual blending in computational invention of cadences and chord progressions in jazz, we model the composition of musical phase spaces as categorical colimits of algebraic specifications. The context in which we consider such colimits is that of service-oriented computing – a paradigm that supports the development of complex software applications based on dynamic reconfigurations of networks of systems (Fiadeiro, 2012). These reconfigurations arise from interactions between software entities, are governed by a need-fulfilment mechanism (software applications connect to external suppliers in order to meet their business goals), and consist of three distinct run-time processes: service discovery, selection and binding. What is particularly important for our work is that the binding of services, which is technically achieved through colimits, can be regarded as the service-oriented counterpart of phase-spaces composition.

The consequences of constraining free jazz

Constraint programming has already proved to be appropriate for computational music composition and modelling music theory disciplines such as harmony, rhythm, instrumentation and counterpoint (Anders and Miranda, 2011). The declarative nature and the modularity of such constraint satisfaction problem (CSP) systems match the way in which composition rules are commonly expressed in standard music theory. Deciding the satisfaction of certain properties or rules based on the true/false dichotomy is however often inadequate for the purpose of composing music, even more so when dealing with improvisation. Soft constraint satisfaction problem systems, which generalise the classical crisp variant of CSP by evaluating constraints over richer truth structures like c-semirings, valuation structures, or residuated lattices, mitigate the problem of expressing loose rules

and provide more flexibility in writing musical guidelines. We enrich the specifications of music phase spaces with the mechanism of soft CSP, considering that each musical section has a set of preferences (soft constraints) that need to be satisfied by the music that will follow it. The values with which the requirements are satisfied are elements of the truth structure's underlying set denoted herein by Sat. The final aim of this soft CSP formalization is to find those specifications of music phase spaces that optimize the satisfaction of the given constraints.

We illustrate the formalization of free-jazz improvisations in the context of service-oriented computing starting from the example presented in (Borgo and Goguen, 2005). The authors analysed an excerpt entitled "Hues of Melanin" from the 1973 Sam Rivers Trio's concert at Yale University. They proposed a sectional interpretation of the performance and highlighted the transitions between the music phases in order to demonstrate the nonlinear dynamics of the improvisation. The segmentation is natural and determined by the frequent and clear variations – rhythmic, timbral or chromatic – of the music flow. We focus on the first part of this examination, namely sections A to H. Although we try to make the presentation of this example self-explanatory, the reader is encouraged to consult the section "Hues of Melanin" of (Borgo and Goguen, 2005).

We consider that each musical section imposes some constraints regarding the tempo, texture, intensity, or technique details of the next musical phase to be played. But we keep in mind that, as the improvisation builds, the constraints of a musical segment evolve and adapt to the already unfolded music: the same musical section or trigger of a transition could require different continuations if played at two different moments of the performance.

spec FREEJAZZ =
sorts Phase, Tempo, Texture, Instrument, Detail, Transition
ops slow, medium, fast : Tempo
repetition, groove, complexity, fragmentation, rubato : Texture
trill, cadence, groove, drone, glissando, ascent, pedal : Detail
N, T1, T2, T3, T4, T5, T6, T7 : Transition
tempo : Phase $\rightarrow$ Tempo
texture : Phase $\rightarrow$ Texture
detail : Phase $\rightarrow$ Detail
transition : Phase $\rightarrow$ Transition

Figure 1: The specification FREEJAZZ

We zoom in on the two transitions triggered by a soprano saxophone trill on D (sections C and G in Figure 2) and we regard the trill as a determining component in the evolution of the improvisation. We model these musical phases as specifications written over first-order logic using a CASL-like syntax (Mosses, 2004), sharing two common sub-specifications: one describing the truth structures used to evaluate the satisfaction of the constraints, i.e. residuated lattices (Galatos et al., 2007), and the other, the specification FREEJAZZ in Figure 1, listing the instruments played by the musicians, possible values for measuring the tempo, texture descriptors, techniques, and ornamentations that constitute the salient details of musical segments, as well as the

Letter	Time	Transition Type	Overall Texture
C	5:29	T2 (soprano trill on D)	FREE
C2	5:48	T7 (drum cadence)	
C3	6:43	T2 (soprano high note)	
C4	7:05	T2/T4/T6 (soprano, bass low A)	
G	13:10	T2 (trill on D), T5 (bass groove)	GROOVE
G2	14:04	metric sync	
G3	14:52	T6 (bass triggers descent)	FREE

Transition types: T2 *pseudo-cadential segue* – an implied cadence with sudden and unexpected continuation; T4 *feature overlap* – one feature of the antecedent section is sustained and becomes part of the consequent section; T5 *feature change* – a gradual change of one feature that redirects the flow (usually subtly); T6 *fragmentation* – a gradual breaking up, or fragmenting, of the general texture and/or rhythm; T7 *internal cadence* – a prepared cadence followed by a short silence then continuation with new material.

Figure 2: Sections and Subsections of "Hues of Melanin" (excerpt from (Borgo and Goguen, 2005), Figure 1)

types of transitions between the sections. Apart from these, a specification describing a musical phase also records the characteristics of the fragment and preferences on the musical section that will continue it. These are expressed as ordinary first-order sentences. In Section "Anthony Braxton Graphic Notation Logic" we make explicit the temporal distinction that separates them into properties of the current music fragment and properties of the next fragment.

spec TRIGGERINGPHASE = FREEJAZZ **and** RESIDUATEDLATTICES
then ops available : Phase $\times$ Instrument $\times$ Instruments $\rightarrow$ Sat
tempoPref : Tempo $\rightarrow$ Sat
texturePref : Texture $\rightarrow$ Sat
instrPref : Instrument $\rightarrow$ Sat
instrumentsPref : Set(Instrument) $\rightarrow$ Sat
detailPref : Detail $\times$ Instrument $\rightarrow$ Sat
transitionPref : Transition $\rightarrow$ Sat
$\forall$ p : Phase; i : Instrument; is : Instruments
• available(p, i, is) = 1 $\Leftrightarrow$ (detail(p) = trill) $\wedge$ (i = sax)
• tempoPref(slow) $\leq$ tempoPref(medium)
• tempoPref(fast) $\leq$ tempoPref(slow)
• texturePref(complexity) $\leq$ texturePref(rubato)
• texturePref(complexity) $\leq$ texturePref(groove)
• instrPref(bass) = instrPref(drums) = instrPref(sax)
• instrPref(flute) $\leq$ instrPref(sax)
• instrumentsPref(S) = $*$(instrPref(S))
• detailPref(trill, sax) = 0
• transitionPref(N) = 0

Figure 3: The specification TRIGGERINGPHASE

For the triggering passage, we can distinguish a number of general constraints that do not depend on the context in which it is played, such as basic guidelines on the tempo, the texture, or the instruments to be played. Even though more specific preferences regarding the details of the section are left to be fixed at the actual moment of the musicking, there are some constraints regarding the need of a transition, and

thus a continuation of the passage – the concert can't end with the trill – and the repetition of the passage – it shouldn't be continued with another saxophone trill. These restrictions are expressed as soft constraint sentences, or *requirements* (here first-order terms), that extend the specification TRIGGERINGPHASE in Figure 3:

cvars phase : Phase; instrument : Instrument;
instruments : Instruments
- tempoPref(tempo(phase))
- texturePref(texture(phase))
- instrumentsPref(instruments)
- transitionPref(transition(phase))
- detailPref(detail(phase), instrument)

These restrict the entire musical phase space to several smaller phase spaces that satisfy the 'needs' of our musical component. In (Borgo and Goguen, 2005), the sections C and G that follow the saxophone trills on D are considered to be alternatives for the development of the performance from our transitional point onwards: we can regard them as different solutions for a constraint problem. Although both sections satisfy the requirements imposed by playing the trill (the sentences of the specification TRIGGERINGPHASE), the valuation of the constraint sentences above (which come with the already performed music) will discriminate between one choice and the other. We can think of sections C and G as epitomes of two growth directions or musical phase spaces: in the first phase, a medium-tempo short bass groove passage is soon abandoned for a rubato (the phase space fights the groove towards a modal area), while in the second section a groove similar to the one that was ended prematurely is explored (the phase space comprises passages of groove exploration and/or increased complexity).

The specifications PHASESPACEC and PHASESPACEG correspond to the two alternative phase spaces presented in our example above. Each of these contains one of the two sections played during the actual improvisation.

spec PHASESPACEC = FREEJAZZ **and** RESIDUATEDLATTICES
then ops available : Phase × Instrument × Instruments → Sat
p1, p2 : Phase
∀ p : Phase; i : Instrument; is : Instruments
- available(p, i, is) = 1 ⇔ (p, i, is) belongs to the following table

ph	tempo	texture	instruments	detail + in	T
p1	medium	rubato	drums, bass, sax	cadence: drums	T7
p2	slow	rubato	drums, bass	cadence: drums	T6

Figure 4: The specification PHASESPACEC

spec PHASESPACEG = FREEJAZZ **and** RESIDUATEDLATTICES
then ops available : Phase × Instrument × Instruments → Sat
p1, p2, p3 : Phase
∀ p : Phase; i : Instrument; is : Instruments
- available(p, i, is) = 1 ⇔ (p, i, is) belongs to the following table

ph	tempo	texture	instruments	detail + in	T
p1	medium	groove	drums, bass, flute	drone: bass	T5
p2	medium	groove	drums, bass, sax	groove: bass	T5
p3	fast	complexity	drums, bass, sax	trill: sax	T2

Figure 5: The specification PHASESPACEG

Service-oriented computing

Building on these specifications of music phases, we will model free-jazz improvisations as service-oriented processes. The framework of service-oriented computing that we consider herein is in keeping with (Chiriţă, Fiadeiro, and Orejas, 2016) and deals with two kinds of entities: *service applications* and *service modules*. The former are the executing units that trigger the discovery of a required service or resource, whereas the modules providing services will be executed only after they are bound to the application. Service applications have an orchestration part, a specification defining their behaviour, and interfaces describing the services required: interfaces are sub-specifications of the given orchestration together with properties that express preferences regarding the use of a given service. Modules are similar to applications in that they comprise an orchestration and interfaces; in addition, they include a description of the functionality or the resources provided.

Definition 1. A *service application* (Σ, I) consists of a specification Σ, called *orchestration*, together with a finite family $I = \{(i_x, r_x)\}_{x \in \overline{n}}$ of *interfaces*, each of which consisting of a mapping $i_x \colon \Sigma_x \to \Sigma$ such that Σ and Σ_x share the same residuated lattice, and a *requirement* $r_x \in \mathrm{CSen}(\Sigma_x)$, where $\mathrm{CSen}(\Sigma_x)$ is the set of all constraint sentences that can be associated to Σ_x.

Definition 2. A *service module* (Ω, P, J) consists of an *orchestration* Ω, a *provides-property* $P \in \mathrm{CSen}(\Omega)$, and a finite family $J = \{(j_y, q_y)\}_{y \in \overline{m}}$ of *interfaces*, consisting of mappings $j_y \colon \Omega_y \to \Omega$ and *requirements* $q_y \in \mathrm{CSen}(\Omega_y)$.

We consider that the execution of service applications takes place in the context of a fixed set of service modules – a *service repository*. Each execution step is triggered by the need to fulfil a requirement of the current application, which in the context of our work corresponds to a requires-interface. Similarly to conventional soft-constraint satisfaction problems, the goal is to maximize the satisfaction of the requirement. To this end, we distinguish three elementary processes: discovery, selection and binding. For a service application and one of its requires-interfaces, the *discovery process* will provide a set of possible matches, i.e. pairs of service modules from the repository and attachment mappings from the requires-specification to the orchestrations of the modules. Note that the output of the discovery process only depends on the repository and the selected requires-interface, and not on the application itself. In the *selection process*, for every match retrieved in the discovery phase, a score of compatibility with the requirement will be computed using the concept of graded semantic consequence (Diaconescu, 2014): the value with which the provides-property semantically entails the requirement. The application then commits to the chosen provider through the *binding process*: the orchestration of the application is blended with the orchestration of the selected service module (via the computation of a pushout of the two specifications), while the fulfilled requirement is replaced with the requirements of the added module. For more technical details on how the service processes are modelled, we refer the interested reader to (Chiriţă, Fiadeiro, and Orejas, 2016).

Music improvising as service-oriented processes

We model a free-jazz performance as a service application, and each musical section to be played as a module. In both cases, the first-order specifications of the music fragments act as orchestrations, while the constraint sentences act as requires-interfaces. Each musical transition determines a round of processes of service discovery, selection and binding: for each section played, a best supplier (musical passage) will be selected from the pool of all possible continuations, and it will be "added" to the current application, thus extending the record of the music already played. Choosing a version or another at a given point in the development of the music depends on the way the selection of a best model is defined, and also on the content of the orchestration at that point: playing a musical trigger will influence both the way we assign a preference value to a musical-fragment candidate – the interpretations of the constraints – and the space of the satisfaction values upon which we judge the compatibility of two musical segments.

Upon the selection of one provider, the binding of the application to the chosen service module represents the commitment of the music performance to one of the two phase spaces: the music blending is realized by the pushout of the two specifications that define the orchestrations.[1] We stress the fact that in this case, the concept blending is not determined by the mere juxtaposition of musical fragments, but by the fact that the class of possible music exploration paths is narrowed with each binding through refinement. At the end of each reconfiguration round, the performed improvisation could be evaluated using two results presented in (Chiriţă, Fiadeiro, and Orejas, 2016): the formalization of the concept of α-satisfiability (where α is a degree of satisfaction) and a theorem stating that the binding process is sound with respect to α-satisfiability.

Modelling "Hues of Melanin"

Consider the service application $\mathcal{A} = (\Sigma, I)$ whose orchestration Σ is TRIGGERINGPHASE (as in Figure 3), and whose interface consists of the identity map and the requirement

$$\mathsf{tempoPref}(\mathsf{tempo}(\mathsf{phase})) \wedge \mathsf{texturePref}(\mathsf{texture}(\mathsf{phase})) \wedge \mathsf{instrumentsPref}(\mathsf{instruments}) \wedge \mathsf{transitionPref}(\mathsf{transition}(\mathsf{phase})) \wedge \mathsf{detailPref}(\mathsf{detail}(\mathsf{phase}), \mathsf{instrument}).$$

We fix the repository $Rep = \{\mathcal{C}, \mathcal{G}\}$, where

- the service module $\mathcal{C} = (\Omega, P, J)$ has the orchestration PHASESPACEC in Figure 4, the provides-property $P = \mathsf{available}(\mathsf{phase}, \mathsf{instrument}, \mathsf{instruments})$, and no requirements, and
- the service module $\mathcal{G} = (\Omega', P', J')$ is such that Ω' is as in Figure 5, and $P' = P$.

[1] Although we use colimits to model concept blending, we are not strictly following the approach from (Goguen, 1999). We do not generate the input morphisms and the base space: the base space and one of the input spaces are part of the service application; the other input space, together with its corresponding morphism, follow from the discovery process, which is modelled here as an external mechanism that can be thought of as a 'black box'.

When selecting a best supplier for $\mathcal{A}$, the music phases that best fit the preferences are phase $p1$ for $\mathcal{C}$ and phase $p2$ for $\mathcal{G}$. In principle, we would need to compute the compatibility scores between TRIGGERINGPHASE and PHASESPACEC and PHASESPACEG, respectively, using all possible models. However, due to the way the specifications are written, the choice of the best phase for each phase space can be inferred directly from the axioms. First, the constraint variables phase, instrument and instruments are limited to the interpretations defined in the tables. Second, the axioms of TRIGGERINGPHASE that express specific preferences, such as for a tempo, make it feasible to determine the best phases provided by each phase space for any model. With respect to tempo, phase $\mathcal{G}.p3$ is the least preferred, while $\mathcal{C}.p1$ and $\mathcal{G}.p1$, $\mathcal{G}.p2$ are the most preferred because $\mathsf{tempoPref(fast)} \leq \mathsf{tempoPref(slow)} \leq \mathsf{tempoPref(medium)}$. However, we cannot decide which one of the two phase spaces would be more suitable, since we cannot decide whether $\mathcal{C}.p1$ or $\mathcal{G}.p2$ satisfies better the constraints. Therefore, any of the two modules could be non-deterministically selected.

Logics for improvising

First-order logic, although standard for formal specifications, may not be the most suitable logic for describing the features of a music fragment; we could specify music by employing other logics with a more convenient syntax, closer to the usual musical notation. To ensure that the framework presented in the previous section can still be used for dealing with improvised music, we impose a set of reasonable restrictions that the new logics must meet. We argue that any logic satisfying the constraints described in (Chiriţă, Fiadeiro, and Orejas, 2016) is generally suitable for capturing free-jazz improvisations: informally, the logic should

- permit the expression of constraints, as the aim of free jazz is "to play together with the greatest possible freedom – which, far from meaning without constraint, actually means to play together with sufficient skill and communication to be able to select proper constraints in the course of the piece"(musician Ann Farber, see (Borgo, 2005), Chapter "Reverence for Uncertainty"),
- permit the partial satisfaction of constraints, as improvisation requires flexibility and non-rigid answers, and
- allow the change of the truth system, as players in a collaborative performance usually have different beliefs and value systems that they impose to the group alternately.

Non-standard notations used in composition notes and guide scores for improvised performances could be used as a basis for defining logics having the expressiveness needed for specifying free jazz. In the following, we show how such notations can be formalized as appropriate logics for improvisation processes, focusing on one of Anthony Braxton's alternative notations for free-jazz composition.

Anthony Braxton Graphic Notation Logic

Through the extensive use of graphic and symbolic notations, Braxton's music positions itself at the fuzzy border between composition and improvisation – see (Lock, 2008).

Neither completely notated, nor completely free, the scores can be seen as an incipient guideline for the improviser: the visual elements force the performer to assign personal interpretations to rather abstract forms that would otherwise make the scores unplayable by immutably following the more conventional notations. The improviser must hence intervene considerably in the composition, not in the usual form of jazz extended solos, but with "tiny pockets of improvisational space" (Lock, 2008) that should fill the non-finished musical structure. This porosity, an inviting-to-improvisation characteristic of his compositions, is also apparent in the work on which we will focus: "Composition 94 for Three Instrumentalists" (Braxton, 1988). In section B of this piece, Braxton uses an *image grouping notation* consisting of three types of contours that are overlaid on top of standard pitches to create the so-called *liquid*, *shape*, and *rigid formations*. The role of the formations is to indicate the performers the outlines that they should follow in playing the notes inside them: these pitches should not be played as they appear in the score, but transformed according to the distinctive interpretation of each improviser. The pitches inside liquid formations, figures resembling clouds, should be played as "clouded mass sound imprints", the shape formations should suggest "harder edges", while the rigid formations, closer to geometrical figures, should highlight their "composite state". In this study, we loosen the restrictions for the interpretation of the formations, blurring the distinction between the three types of formations, and we accept as valid the improvisations that replace the notes within the shapes with completely different pitches given that they both allude to the original notes, and evoke the contours. We will reduce the problem of quantifying the improvisation's reminiscence of the original pitches to the problem of measuring the similarity of two music fragments (the interested reader is referred, for example, to (Mongeau and Sankoff, 1990) for further details on the comparison of musical fragments).

The observations above lead to a straightforward formalization of Braxton's graphic notation as a many-valued logic $\mathcal{BGN}$. To obtain a representation of the scores, and furthermore, to express properties of the music segments at certain positions in a score, the language of $\mathcal{BGN}$ must comprise the universe of all possible music fragments written in a standard notation, a set of formations, and an appropriate truth structure that will allow us to manipulate partially true statements. We define hence a signature Σ as a triple $(\mathcal{L}, \mathrm{MF}, \mathrm{FS})$, where $\mathcal{L}$ is a residuated lattice, MF is a set of music fragments, and FS is a set of formation symbols. The morphisms of signatures, i.e. mappings that permit translations from one language to another, are defined componentwise: a morphism $\varphi\colon \Sigma \to \Sigma'$ consists of a morphism of residuated lattices $\varphi_{rl}\colon \mathcal{L}' \to \mathcal{L}$, a function φ_{MF} between the sets of fragments MF and MF′, a function φ_{FS} from the set of formation symbols FS to FS′, and a natural number l representing a delay between the moment of playing a score written over the first signature and the moment of playing a score written over the second one. We admit three types of atomic sentences built using the symbols in the signatures:

- $m@p$, with $p \in \mathbb{N}$ and $m \in \mathrm{MF}$, which should be read as "at position p we have the music fragment m",
- $\sim m@p$, with $p \in \mathbb{N}$ and $m \in \mathrm{MF}$, which should be read as "at position p we have a fragment similar with m", and
- $s(@p)$, with $s \in \mathrm{FS}$ and $p \in \mathbb{N}$, which should be read as "the fragment at position p is in the shape of s".

We will denote by $s_p(m)$ the conjunction $s(@p) \wedge \sim m@p$.

For any morphism of signatures $\varphi\colon \Sigma \to \Sigma'$, we can translate the atoms over Σ to atoms over Σ' as follows:

- $\varphi(m@p) = (m@(p+l))$
- $\varphi(\sim m@p) = \sim m@(p+l)$
- $\varphi(s(@p)) = \varphi_{\mathrm{FS}}(s)(@p+l)$

The semantics of $\mathcal{BGN}$ is given by classes of models corresponding to every signature: every Σ-model M consists of a set $|M|$ of interpretations of music fragments, a method $M_\sim\colon |M| \times |M| \to \mathcal{L}$ for measuring the similarity of two segments as a value of $\mathcal{L}$, interpretations $M_s\colon |M| \to \mathcal{L}$ of the formation symbols $s \in \mathrm{FS}$, and a sequence of music fragments $M_{\mathrm{seq}} \in |M|^*$ to be played. We will usually denote the fragment at the position p in M_{seq} by $M_{\mathrm{seq}}[p]$; we will sometimes describe a sequence through a regular-expression-like string in which interrogation points mark the parts that are yet to be fixed (containing formation symbols), and the $*$ symbol marks the fact that the sequence is open and admits any possible succession of music fragments.

Models M can satisfy a sentence ρ with a many-valued truth degree from the residuated lattice, denoted by $M \models \rho$:

- $M \models (m@p)$ is defined as $\begin{cases} 0, & \text{if } M_{\mathrm{seq}}[p] \neq m \\ 1, & \text{if } M_{\mathrm{seq}}[p] = m \end{cases}$,
- $M \models (\sim m@p)$ is given by the similarity of m and the music fragment at position p, i.e. $M_\sim(m, M_{\mathrm{seq}}[p])$,
- $M \models (s(@p))$ is given by the resemblance $M_s(M_{\mathrm{seq}}[p])$ of the fragment at position p with the shape s.

The fact that a signature does not determine the interpretation of the music fragments, the similarity measure, or the interpretation of the formation symbols, makes the logic $\mathcal{BGN}$ too general for suitably specifying music: we would like to be able to control, for example, which similarity measure to use in comparing music fragments. We hence define $\mathcal{SBGN}$ to be the logic having as signatures pairs consisting of $\mathcal{BGN}$-signatures Σ and fixed classes $\mathcal{M}$ of Σ-models. The signature morphisms of $\mathcal{SBGN}$, which will play an important role in defining service discovery and binding, are defined as usual, as the $\mathcal{BGN}$-signature morphisms $\varphi\colon \Sigma \to \Sigma'$ for which the associated model reduct $\varphi_{\mathcal{M}}$[2] satisfies the property $\varphi_{\mathcal{M}}(\mathcal{M}') \subseteq \mathcal{M}$.

A "Clapping Music" improvisation

To illustrate how an improvised performance based on composition notes written using the graphic notation of Anthony Braxton can be seen as a series of dynamic processes between service modules specified over $\mathcal{SBGN}$, we choose to

[2] We recall that any Σ'-model M' can be reduced along the signature morphism φ to a Σ-model $\varphi_{\mathcal{M}}(M')$ simply by forgetting the interpretations of the new symbols that the morphism introduces; see (Sannella and Tarlecki, 2012) for more details.

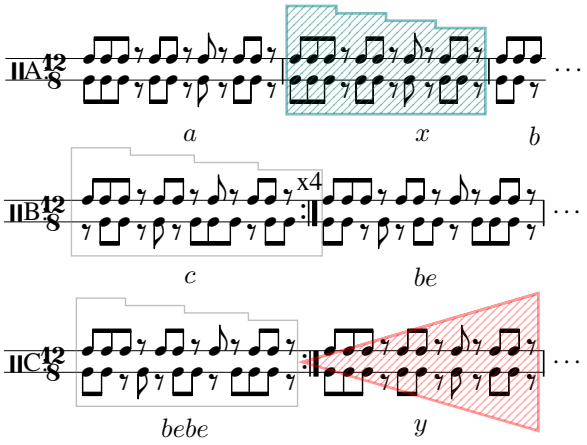

Figure 6: Scores A, B, C

reduce to a minimum the details particular to music-theory. This simplification is intended to: (1) underline the fact that the freeness of the performance does not reside primarily in the qualitative aspects of the resulted music, but in the nature of the musicking process itself, and (2) alleviate the effort of the reader unfamiliar with basic notions of music theory.

We formalize our example starting from Steve Reich's minimalist composition "Clapping Music" written in 1972. Although a complete composition, with no musical segments meant to be improvised, the piece constitutes a good reference for our purpose due to its simplicity. Written around a basic pattern very similar to the standard African 12/8 bell pattern, the piece should be played by two performers: one should continuously and unvaryingly repeat the basic pattern, while the other should repeat and shift the pattern with one note after each eight bars.

We use fragments of this composition to exemplify the binding of services that specify incomplete musical segments written in Braxton's notation: we start from the first bar of the piece, the basic pattern (see Figure 6, A), and we let the performance develop according to both fixed, rigid instructions, and loose, subject to improvisation guidelines. Bar a is followed by a shape formation $\boxdot$ specifying that the pattern should be repeated, but in a transformed state reminding of descending steps (fragment x), and by the fixed fragment b, to which other fragments may be added.

In the following, we will consider the universe MF of musical fragments to be the set of all the possible score fragments obtained from composing the basic pattern a and the patterns obtained by shifting it, together with the prefixes of these shifts. We formalize the starting score as a service application $\mathcal{A} = (\Sigma, I)$ with

- the orchestration Σ given by $(\mathcal{L}_\Sigma, \mathrm{MF}, \{\boxdot\}, \mathcal{M}_\Sigma)$, where $\mathcal{M}_\Sigma$ is the class of the models that correspond to sequences of music fragments described as $a?b*$;
- a single interface $(i\colon \Sigma_1 \to \Sigma, R)$, where i_{rl}, i_{MF} and i_{FS} are all identities, $i_l = 1$ (to indicate that the variable fragment x appears at position 1) and $i_{\mathcal{M}}$ the inclusion of $\mathcal{M}_\Sigma$ in the class $\mathcal{M}'_\Sigma$ of models that correspond to all sequences of music fragments, which are described as $*$,[3] and $R = \boxdot_0(x) = \boxdot(@0) \wedge (\sim x@0)$.

To refine and continue the given music score, we consider a round of processes of discovery, selection and binding of other music fragments to our original fragment. Let the result of the discovery process be the set of the scores B and C in Figure 6. Formally, they are service modules $\mathcal{B} = (\Omega_\mathcal{B}, P_\mathcal{B}, J_\mathcal{B})$ and $\mathcal{C} = (\Omega_\mathcal{C}, P_\mathcal{C}, J_\mathcal{C})$ as follows:

- their orchestrations are $\Omega_\mathcal{B} = (\mathcal{L}_{\Omega_\mathcal{B}}, \mathrm{MF}, \emptyset, \mathcal{M}_{\Omega_\mathcal{B}})$ and $\Omega_\mathcal{C} = (\mathcal{L}_{\Omega_\mathcal{C}}, \mathrm{MF}, \{\lhd\}, \mathcal{M}_{\Omega_\mathcal{C}})$ with the classes of models $\mathcal{M}_{\Omega_\mathcal{B}}$ and $\mathcal{M}_{\Omega_\mathcal{C}}$ defined by the sequences of music fragments $cbe*$ and $bebe?*$, respectively;
- they guarantee to begin with the fragments c and $bebe$[4] through the provides-properties $P_\mathcal{B} = c@0$ and $P_\mathcal{C} = bebe@0$;
- we model the fact that the score B is completely fixed, not demanding improvisation, by considering the set of requirements to be empty;
- the interface $j_\mathcal{C}\colon \Omega'_\mathcal{C} \to \Omega_\mathcal{C}$ of $\mathcal{C}$ is defined similarly to the interface of $\mathcal{A}$: it consists of identities for the residuated lattice, the music fragment space and the set of formation symbols, the natural number 1, indicating the position of the formation symbol in the score, and the inclusion of the class of models $\mathcal{M}_{\Omega_\mathcal{C}}$ in the class $\mathcal{M}_{\Omega'_\mathcal{C}}$ of models having all possible sequences of music fragments;
- the requirement $Q_\mathcal{C} = \lhd_0(y) = \lhd(@0) \wedge (\sim y@0)$.

In order to perform a selection between the two service modules, we would have to further limit the classes of models $\mathcal{M}_\Sigma$, $\mathcal{M}_{\Omega_\mathcal{B}}$ and $\mathcal{M}_{\Omega_\mathcal{C}}$. To determine how suitable one fragment is for the intended improvisation, we would need to fix a set of measures of similarity between music fragments and of acceptable interpretations for the formation symbol $\boxdot$: How similar are the fragments c and $bebe$ to x, and how much can they be perceived as sounds in the shape of $\boxdot$? Let us skip this phase of our running example, considering the score B to be the result of an arbitrary selection process, and focus on the binding of modules as a process of blending music fragments.

By computing the pushout of the morphisms i and ϕ that map the requires-specification to the orchestration of the application and to the provides-specification of the service module, we amalgamate the models[5] in $\mathcal{M}_\Sigma$ and $\mathcal{M}_{\Omega_\mathcal{B}}$, and hence their musical sequences, $a?b*$ and $cbe*$ respectively, obtaining the contiguous music score $acbe*$. (Note the role of the delay 1 corresponding to the morphisms i and j.)

$$\begin{array}{ccc} \Sigma_1 & \xrightarrow{\ i\ } & \Sigma \\ {\scriptstyle\phi}\downarrow & po & \downarrow{\scriptstyle i'} \\ \Omega_\mathcal{B} & \xrightarrow[\ j\ ]{} & \Sigma' \end{array} \qquad \begin{array}{ccc} * & \xrightarrow{\ 1\ } & a?b* \\ {\scriptstyle 0}\downarrow & & \downarrow{\scriptstyle 0} \\ cbe* & \xrightarrow[\ 1\ ]{} & acbe* \end{array}$$

[3] We choose not to limit the class of models through the interface because we want to be able to consider as a candidate in the selection process any music fragment satisfying the requirement R.

[4] Note the $\times 4$ superscript in Figure 6, B denoting that c is the repetition of the highlighted fragment four times.

[5] Here we use the specification-theoretic notion of model amalgamation, see for example (Sannella and Tarlecki, 2012).

The substitution of the orchestration Σ of $\mathcal{A}$ with the vertex Σ' of the pushout will hence determine the refinement of the class of models $\mathcal{M}_\Sigma$, both filling the improvisational gaps of the performance, and molding its evolution.

Conclusions

In this paper we have shown how the fundamental processes of service discovery, selection and binding can be used to model free-jazz performances. The main step in this endeavour was to identify which logical formalisms are suitable for capturing and reasoning about free-jazz, and at the same time are compatible with the principles of the service-oriented computing paradigm. To this end, we have first discussed a variant of many-valued first-order logic endowed with constraints, which is closer to the formalisms used in previous developments on service-oriented computing (Chiriţă, Fiadeiro, and Orejas, 2016); we have then defined a novel logic, particular to free-jazz, based on Anthony Braxton's graphic notations. The proposed formalization paves the way to reasoning about improvisation processes: one can now study aspects related to reliability (to what extend the user's expectations can be met?), determine which music fragments are hardly reachable (never or seldom used during a play), or make predictions about the evolution of an improvisation (e.g. how does the choice of a music fragment affect the subsequent use of other fragments?).

Our work has focused on the musicking process itself, not on the resulting music: we do not provide a way to record the improvisation; instead, the music is played at run-time, whenever a new fragment is sublated into the performance. We would like to continue to pursue this line of research by implementing an $\mathcal{SBGN}$ specification and programming language whose operational semantics extends the logic programming of services from (Ţuţu and Fiadeiro, 2015) by taking into account many-valued truth spaces. In this way, we could create novel computationally creative music systems that exploit the operational semantics of service applications to deliver improvised music to users based on an online repository of music fragments. The repository would be ever-changing, hence, even if the user's input were the same, the composed music could vary significantly.

Acknowledgements

The authors are grateful to Nuno Barreiro for his continuous encouragement and helpful discussions throughout the course of this work.

References

Anders, T., and Miranda, E. R. 2011. Constraint programming systems for modeling music theories and composition. *ACM Comput. Surv.* 43(4):30.

Blackwell, T., and Young, M. 2004. Self-organised music. *Organised Sound* 9(02):123–136.

Borgo, D., and Goguen, J. 2005. Rivers of consciousness: The nonlinear dynamics of free jazz. In *Jazz Research Proceedings Yearbook*, volume 25.

Borgo, D. 2005. *Sync or swarm: Improvising music in a complex age*. A&C Black.

Braxton, A. 1988. *Composition notes*, volume 3. Synthesis Music.

Chiriţă, C. E.; Fiadeiro, J. L.; and Orejas, F. 2016. Many-valued institutions for constraint specification. In *FASE 2016*, volume 9633 of *LNCS*, 359–376. Springer.

Diaconescu, R. 2014. Graded consequence: an institution theoretic study. *Soft Comput.* 18(7):1247–1267.

Eppe, M.; Confalonieri, R.; Maclean, E.; Kaliakatsos-Papakostas, M. A.; Cambouropoulos, E.; Schorlemmer, W. M.; Codescu, M.; and Kühnberger, K. 2015. Computational invention of cadences and chord progressions by conceptual chord-blending. In *IJCAI 2015*, 2445–2451. AAAI Press.

Fiadeiro, J. L. 2012. The many faces of complexity in software design. In *Conquering Complexity*. Springer. 3–47.

Galatos, N.; Jipsen, P.; Kowalski, T.; and Ono, H. 2007. *Residuated Lattices: An Algebraic Glimpse at Substructural Logics*. Studies in Logic and the Foundations of Mathematics. Elsevier Science.

Goguen, J. 1999. An introduction to algebraic semiotics, with application to user interface design. In *Computation for metaphors, analogy, and agents*. Springer. 242–291.

Kaliakatsos-Papakostas, M.; Cambouropoulos, E.; Kühnberger, K.-U.; Kutz, O.; and Smaill, A. 2014. Concept invention and music: Creating novel harmonies via conceptual blending. In *CIM2014*.

Lock, G. 2008. What I call a sound: Anthony Braxton's synaesthetic ideal and notations for improvisers. *Critical Studies in Improvisation* 4(1).

Mazzola, G., and Cherlin, P. B. 2008. *Flow, gesture, and spaces in free jazz: Towards a theory of collaboration*. Springer Science & Business Media.

Mongeau, M., and Sankoff, D. 1990. Comparison of musical sequences. *Comput. Hum.* 24(3):161–175.

Mosses, P. D. 2004. *CASL Reference Manual*, volume 2960 of *LNCS*. Springer.

Pearce, M.; Meredith, D.; and Wiggins, G. 2002. Motivations and methodologies for automation of the compositional process. *Musicae Scientiae* 6(2):119–147.

Ramalho, G., and Ganascia, J. 1994. Simulating creativity in jazz performance. In *AAAI 1994*, 108–113. AAAI Press / The MIT Press.

Sannella, D., and Tarlecki, A. 2012. *Foundations of Algebraic Specification and Formal Software Development*. Springer.

Ţuţu, I., and Fiadeiro, J. L. 2015. Service-oriented logic programming. *LMCS* 11(3):1–38.

Wiggins, G. A.; Pearce, M. T.; Müllensiefen, D.; et al. 2009. Computational modeling of music cognition and musical creativity. In *Oxford Handbook of Computer Music*. Oxford University Press. chapter 19.