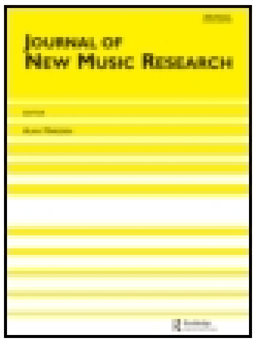

Journal of New Music Research

ISSN: 0929-8215 (Print) 1744-5027 (Online) Journal homepage: http://www.tandfonline.com/loi/nnmr20

Musical agents: A typology and state of the art towards Musical Metacreation

Kıvanç Tatar & Philippe Pasquier

To cite this article: Kıvanç Tatar & Philippe Pasquier (2018): Musical agents: A typology and state of the art towards Musical Metacreation, Journal of New Music Research, DOI: 10.1080/09298215.2018.1511736

To link to this article: https://doi.org/10.1080/09298215.2018.1511736

Published online: 10 Sep 2018.

JOURNAL OF NEW MUSIC RESEARCH
https://doi.org/10.1080/09298215.2018.1511736

Musical agents: A typology and state of the art towards Musical Metacreation

Kıvanç Tatar and Philippe Pasquier

Interactive Arts and Technology, Simon Fraser University, Surrey, Canada

ABSTRACT
Musical agents are artificial agents that tackle musical creative tasks, partially or completely. This review of musical agents combines the terminology of Generative Arts (artistic practice) and the scientific literature of Computational Creativity, Multi-Agent Systems (MAS), and Artificial Intelligence. We define Musical Metacreation as a field that studies the partial or complete automation of musical tasks. We survey seventy-eight musical agent systems, and present a typology of musical agents. After examining the evaluation methodologies of musical agents, we propose possible future steps while mentioning ongoing discussions in the field.

ARTICLE HISTORY
Received 11 December 2017
Accepted 3 August 2018

KEYWORDS
Musical agents; Multi-Agent Systems; Artificial Intelligence; Musical Metacreation; Computational Creativity

1. Introduction

The works of Generative Music rely on autonomous systems for part or all of their production. One type of such systems is the automaton, a self-operating machine that carries out pre-defined procedures. The first musical automaton, al-Jazari's water clock, appeared in the seventh century as a result of advances in hydraulics (Fowler, 1967). This water clock could generate music using a mechanical and hydraulic system. With the onset of the industrial revolution and following the invention of electricity, automatic musical machines entered a new phase in their development, and more musical automata emerged. These included amongst other the Regina Concert Orchestrion, the Autophone and the Link Orchestrion. And now, after the digital revolution, artificial agents have become the modern day equivalent of the automaton.

In the digital age, new autonomous tools and systems have been emerging in creative applications. Artificial Intelligence (AI) and Multi-Agent Systems (MAS) are two examples of fields that provide such autonomous tools. Simon (1960) defines AI as 'the science of having machines solve problems that do require intelligence when solved by humans. MAS are distributed/concurrent systems that are autonomous, able to make independent decisions, and run online (Wooldridge, 2009). Software agents in MAS are autonomous pieces of software which contain perception and action abilities. Applications of MAS are beneficial to modelling and designing musical creativity because musical creativity involves distributed, coordinated entities with perception and action abilities. For example, in a live music performance, musicians collaboratively create music by listening to each other. Similarly, we could distribute composition tasks into such sub-tasks as producing individual instrument parts or layers in experimental music (Roads, 2015).

Our review gives an introduction to researchers and practitioners who are interested in musical agents. Musical agents are artificial agents that tackle musical creative tasks, in part or as a whole, and use the methods of MAS and Artificial Intelligence to automatise these tasks. Thus, this topic is naturally interdisciplinary, combining music, science, design and technology. In this paper, we present a state of the art in musical agents that utilise MAS technologies for musical creativity. Three types of artificial agents appear in the literature: software agents that are purely computational; virtual agents that are embodied in a Computer Generated Image (CGI);[1] and robotic agents that hold a physical form.[2] In our survey of

[1] Please refer to Churchill, and Prevost et al. (2000) and Hartholt et al. (2013) for an introduction to virtual agents in CGI.
[2] Bretan Weinberg (2016) present a state of the art in musically creative robotic agents.

CONTACT Kıvanç Tatar ktatar@sfu.ca Interactive Arts and Technology, Simon Fraser University, 250-13450 102 Avenue, Surrey BCV3T0A3, Canada

Table 1. Musical Agents.

#	System	Architecture	# of Agents	# of roles	Environment	Corpus	Input	Output	Communication	HIM	MuMe Task	Evaluation	Code	Public	Section
	Cognitive Musical Agents														**4**
①	VMMAS	knowledge representation	Multi-agent/ Heterogenous	Multi-role	Real-world	Symbolic	Symbolic	Symbolic	Env.	P	Comp., Accomp.	✓	Not shared		4.1
②	Inmamusys	knowledge representation	Multi-agent/ Heterogenous	Multi-role	Real-world	Symbolic	Symbolic	Symbolic	Mess.	C	Comp., Accomp.	✓	Not shared		4.1
③	Generating Affect	knowledge representation	Multi-agent/ Heterogenous	Multi-role	Real-world	Hybrid	None	Audio	Mess.	–	Comp.		Shared	✓	4.1
④	Coming Together	BDI	Multi-agent/ Homogenous	Single-role	Real-world	–	Symbolic	Hybrid	Hybrid	–	Comp.		Shared	✓	4.2
⑤	Indifference Engine	BDI	Multi-agent/ Heterogenous	Single-role	Hybrid	–	Audio	Audio	Hybrid	P	Improv., Comp.		Not shared	✓	4.2
⑥	MUSIC-MAS	BDI	Multi-agent/ Homogenous	Multi-role	Real-world	Symbolic	Symbolic	Symbolic	Hybrid	C	Assisted Comp., Style Im.	✓	Not shared		4.2
⑦	HSMM	Cognitive	Mono-agent	Single-role	Real-world	Symbolic	Symbolic	Symbolic	Env.	P+C+L	Comp., Assisted Comp.,, Cont.		Not shared	✓	4.3
⑧	MusiCOG	Cognitive	Multi-agent/ Heterogenous		Real-world	Symbolic	Symbolic	Symbolic	Env.	P+L	Comp., Assisted Comp.		Shared	✓	4.3
⑨	MAMA	Cognitive	Multi-agent/ Homogenous	Single-role	Real-world	Hybrid	Hybrid	Hybrid	Hybrid	–	Accomp., Improv.	✓	Shared		4.3
	Reactive Musical Agents	in Real-World Environments													**5.1**
⑩	Cypher	Rule-based	Multi-agent/ Heterogenous	Multi-role	Real-world	Symbolic	Symbolic	Symbolic	Mess.	C	Comp.		Not shared		5.1.1
⑪	Voyager	Rule-based	Multi-agent/ Heterogenous	Single-role	Real-world	–	Hybrid	Symbolic	Env.	P	Improv.		Shared	✓	5.1.1
⑫	Bob	Rule-based	Mono-agent	Single-role	Real-world	Symbolic	Symbolic	Symbolic	Env.	P+L	Improv.,Melody Gen.		Not shared		5.1.1
⑬	ARHS	Rule-based	Multi-agent/ Homogenous	Single-role	Real-world	–	Audio	Audio	Env.	P	Improv.		Not shared	✓	5.1.1
⑭	LL:	Rule-based	Multi-agent/ Homogenous	Single-role	Real-world	–	Audio	Audio	Env.	P	Improv.	✓	Not shared	✓	5.1.1
⑮	Virtualband	Rule-based	Multi-agent/ Heterogenous	Multi-role	Real-world	Hybrid	Audio	Audio	Mess.	P+L	Style Im., Accomp.		Not shared	✓	5.1.1
⑯	Odessa	Rule-based	Mono-agent	Single-role	Real-world	–	Audio	Symbolic	Env.	P	Improv.	✓	Not shared	✓	5.1.1
⑰	Rhythms as···	Rule-based	Multi-agent/ Homogenous	Single-role	Real-world	–	Symbolic	Symbolic	Mess.	–	Rhythm Gen.		Not shared		5.1.1
⑱	VirtuaLatin	Rule-based	Multi-agent/ Heterogenous	Multi-role	Real-world	Symbolic	Symbolic	Symbolic	Env.	P	Rhythm Gen.		Not shared		5.1.1
⑲	DrumTrack	Rule-based	Mono-agent	Single-role	Real-world	–	Audio	Audio	Env.	P	Accomp., Rhythm Gen., Improv.		Not shared	✓	5.1.1
⑳	BBCut2	Rule-based	Mono-agent	Single-role	Real-world	Audio	Audio	Audio	Env.	P+L	Accomp., Rhythm Gen., Improv.		Shared		5.1.1

21	Kinetic Engine	Rule-based/EC	Multi-agent/ Heterogenous	Multi-role	Real-world	–	Symbolic	Symbolic	Mess.	C	Rhythm Gen.		Not shared	✓	5.1.1
22	Beatbender	Rule-based	Multi-agent/ Homogenous	Single-role	Real-world	–	–	Audio	Mess.	–	Rhythm Gen.	✓	Not shared		5.1.1
23	Andante	Rule-based	Multi-agent/ Heterogenous		Real-world		–	Audio	Mess.	–	Comp.		Not shared		5.1.1
24	PIWeCS	Rule-based	Multi-agent/ Heterogenous	Multi-role	Real-world	Audio	Audio	Audio	Hybrid	C	Comp.		Not shared		5.1.1
25	CT: Freesound	Rule-based	Multi-agent/ Homogenous	Single-role	Real-world	Hybrid	Audio	Audio	Hybrid	–	Comp.	✓	Shared	✓	5.1.1
26	Curatorial···	Rule-based	Multi-agent/ Heterogenous	Multi-role	Real-world	Symbolic	Symbolic	Symbolic	–	–	Curation, Comp.		Shared	✓	5.1.1
27	ParamBOT	Rule-based	Multi-agent/ Heterogenous	Multi-role	Real-world	Agents	–	Audio	Mess.	–	Curation, Comp.		Shared	✓	5.1.1
28	GenJam	Evolutionary Computation	Mono-agent	Single-role	Real-world	Symbolic	Symbolic	Symbolic	Hybrid	P+L	Improv.,Melody Gen.		Not shared		5.1.2
29	automated···	Evolutionary Computation	Mono-agent	Single-role	Real-world	–	Audio	Audio	Env.	P	Improv.		Shared		5.1.2
30	Frank	Evolutionary Computation	Mono-agent	Single-role	Real-world	Hybrid	Audio	Audio	Env.	P+L	Improv.		Not shared	✓	5.1.2
31	RGeme	Evolutionary Computation	Multi-agent/ Homogenous	Single-role	Real-world	Symbolic		Symbolic		–	Rhythm Gen.		Not shared		5.1.2
32	··· Tuning···	Evolutionary Computation	Multi-agent/ Homogenous	Single-role	Real-world	–	Symbolic	Symbolic	Hybrid	–	Assisted Comp.	✓	Not shared		5.1.2
		in Virtual Environments													**5.2**
33	Frankensteinian···	Evolutionary Computation	Multi-agent/ Heterogenous	Multi-role	Real-world	Symbolic	–	Symbolic	Hybrid	–	Comp., Assisted Comp.		Not shared		5.2.1
34	Living Melodies	Evolutionary Computation	Multi-agent/ Homogenous	Single-role	Virtual ecosystem	Symbolic	–	Symbolic	Hybrid	–	Comp., Assisted Comp.		Not shared		5.2.1
35	Emergent···	Evolutionary Computation	Multi-agent/ Homogenous	Multi-role	Virtual ecosystem	Symbolic	Symbolic	Symbolic	Env.	–	Rhythm Gen.		Not shared		5.2.1
36	IMAP	Evolutionary Computation	Multi-agent/ Homogenous	Multi-role	Real-world	Symbolic	Symbolic	Symbolic	Env.	–	Interpretation	✓	Not shared		5.2.1
37	RiverWave	Evolutionary Computation	Multi-agent/ Homogenous	Single-role	Virtual ecosystem	–	–	Output	Env.	–	Comp.		Not shared	✓	5.2.1
38	Petri	Evolutionary Computation	Multi-agent/ Homogenous	Single-role	Virtual ecosystem	–	Computer Vision	Audio	Env.	C	Comp.		Not shared	✓	5.2.1
39	Swarm Music	Ecosystemic	Multi-agent/ Homogenous	Single-role	Virtual ecosystem	–	Symbolic	Symbolic	Env.	P	Comp.		Not shared	✓	5.2.2
40	Swarm Granulator	Ecosystemic	Multi-agent/ Homogenous	Single-role	Virtual ecosystem	–	Audio	Audio	Env.	P	Comp.		Not shared	✓	5.2.2
41	Real-time···	Ecosystemic	Multi-agent/ Heterogenous		Virtual ecosystem	–	Hybrid	Hybrid	Env.	P	Improv.		Not shared		5.2.2

(*continued*).

Table 1. Continued.

#	System	Architecture	# of Agents	# of roles	Environment	Corpus	Input	Output	Communication	HIM	MuMe Task	Evaluation	Code	Public	Section
(42)	Nodal	Ecosystemic	Multi-agent/ Homogenous	Single-role	Virtual ecosystem	–	Symbolic	Symbolic	Mess.	C	Comp.		Shared	✓	5.2.2
(43)	OSCAR	Ecosystemic	Multi-agent/ Homogenous	Single-role	Virtual ecosystem	–	Symbolic	Symbolic	Env.	–	Comp.		Shared	✓	5.2.2
(44)	CT: Shoals	Ecosystemic	Multi-agent/ Homogenous	Single-role	Virtual ecosystem	Audio	Parameter	Parameter	Mess.	–	Comp.		Not shared	✓	5.2.2
(45)	earGram Actors	Ecosystemic	Multi-agent/ Homogenous	Single-role	Virtual ecosystem	Audio	–	Audio	Env.	C	Comp.		Not shared	✓	5.2.2
(46)	pMIMACS	Ecosystemic	Multi-agent/ Homogenous	Multi-role	Virtual ecosystem	–	Symbolic	Symbolic	Env.	–	Interpretation	✓	Not shared		5.2.2
(47)	SDS	Ecosystemic	Multi-agent/ Homogenous	Single-role	Virtual ecosystem	–	Symbolic	Symbolic	Mess.	–	Melody Gen.		Not shared		5.2.2
(48)	iMe	Ecosystemic	Multi-agent/ Homogenous	Multi-role	Virtual ecosystem	–	Symbolic	Symbolic	Env.	P	Comp., Assisted Comp.		Not shared		5.2.2
	Hybrid Musical Agents														**6**
(49)	POMDP	Statistical Sequence Modelling	Mono-agent	Single-role	Real-world	–	Symbolic	Symbolic	Env.	P+L	Improv., Style Im.		Not shared		6.1
(50)	Continuator	Statistical Sequence Modelling	Mono-agent	Single-role	Real-world	–	Symbolic	Symbolic	Env.	P+L	Improv., Style Im., Accomp.		Not shared	✓	6.1
(51)	Beatback	Statistical Sequence Modelling	Multi-agent/ Homogeneous	Single-role	Real-world	–	Symbolic	Audio	Mess.	P+C	Rhythm Gen.	✓	Not shared		6.1
(52)	Ringomatic	Statistical Sequence Modelling	Mono-agent	Single-role	Real-world	Hybrid	Symbolic	Audio	Env.	P	Rhythm Gen.	✓	Not shared		6.1
(53)	Using FO···	Statistical Sequence Modelling	Mono-agent	Single-role		–	Symbolic	Symbolic	Env.	P+L	Improv., Style Im.		Not shared	✓	6.1
(54)	OMAX	Statistical Sequence Modelling	Mono-agent	Single-role	Real-world	Hybrid	Hybrid	Hybrid	Mess.	P+L	Improv., Style Im.		Not shared	✓	6.1
(55)	Anticipatory···	Statistical Sequence Modelling	Mono-agent	Single-role	Real-world	Symbolic	Symbolic	Symbolic	Env.	P+C+L	Improv., Style Im.		Not shared		6.1
(56)	Improvagent	Statistical Sequence Modelling	Mono-agent	Single-role	Real-world	Symbolic	Symbolic	Symbolic	Env.	P	Improv.		Not shared		6.1
(57)	Improtek	Statistical Sequence Modelling	Multi-agent/ Heterogenous	Multi-role	Real-world	Hybrid	Hybrid	Hybrid	Env.	P+C+L	Improv., Style Im.		Not shared	✓	6.1

58	AO	Statistical Sequence Modelling	Mono-agent	Single-role	Real-world	Audio	Audio	Audio	Env.	P+C+L	Improv., Style Im.		Not shared	✓	6.1
59	PyOracle	Statistical Sequence Modelling	Mono-agent	Single-role	Real-world	Audio	Audio	Audio	Env.	P	Improv., Style Im.	✓	Shared	✓	6.1
60	VMO	Statistical Sequence Modelling	Mono-agent	Single-role	Real-world	Audio	Audio	Audio	Env.	P+C+L	Improv., Style Im		Not shared	✓	6.1
61	Filter	Statistical Sequence Modelling	Mono-agent	Single-role	Real-world	Hybrid	Audio	Audio	Env.	P	Improv.		Not shared	✓	6.1
62	SpeakeSystem	Statistical Sequence Modelling	Mono-agent	Single-role	Real-world	Symbolic	Audio	Audio	Env.	P+L	Improv.	✓	Shared	✓	6.1
63	ADTK	Statistical Sequence Modelling	Multi-agent/ Heterogenous	Multi-role	Real-world	–	Symbolic	Symbolic	Hybrid	C	Style Im., Improv.		Not shared		6.2
64	CinBalada	Statistical Sequence Modelling	Multi-agent/ Homogenous	Multi-role	Real-world	–	–	Symbolic	Mess.	–	Rhythm Gen.	✓	Not shared		6.2
65	Reactive Accompanist	Artificial Neural Networks	Mono-agent	Single-role	Real-world	–	Audio	Symbolic	Env.	P	Improv., Accomp.		Shared		6.3
66	NN music	Artificial Neural Networks	Mono-agent	Single-role	Real-world	–	Audio	Audio	Env.	P+L	Improv.		Not shared	✓	6.3
67	··· Live Algorithms	Artificial Neural Networks	Mono-agent	Single-role	Real-world	–	Audio	Audio	Env.	P	Improv.		Shared	✓	6.3
68	··· Automated···	Artificial Neural Networks	Mono-agent	Single-role	Real-world	Symbolic	Symbolic	Symbolic	Env.	L	Improv.,Melody Gen.		Not shared		6.3
69	ML.*	Artificial Neural Networks	Mono-agent	Single-role	Real-world	Hybrid	Audio	Audio	Env.	P+L	Improv.		Not shared		6.3
70	Connectionist···	Artificial Neural Networks	Mono-agent	Single-role	Real-world	Symbolic	Symbolic	Symbolic	Env.	–	Rhythm Gen.		Not shared		6.3
71	HARP	Cognitive	Mono-agent	Single-role	Real-world	Symbolic	Hybrid	Hybrid	Mess.	C	Assisted Comp., Improv.		Not shared	✓	6.4
72	Jambot	Cognitive	Mono-agent	Single-role	Real-world	–	Symbolic	Symbolic	Env.	P	Rhythm Gen., Improv., Accomp.		Not shared		6.4
73	··· Motivation···	Cognitive	Mono-agent	Single-role	Real-world	Symbolic	Symbolic	Symbolic	Env.	P	Improv.		Partially shared	✓	6.4
74	Mockingbird	Cognitive	Mono-agent	Single-role	Real-world	Hybrid	Audio	Audio	Env.	P+L	Accomp., Improv.		Not shared		6.4
75	MAgentA	Cognitive	Mono-agent	Single-role	Real-world	Symbolic	–	Symbolic	Mess.	–	Comp.		Not shared		6.4
76	FO with flow	Cognitive	Mono-agent	Single-role	Real-world	Symbolic	Symbolic	Symbolic	Env.	P+L	Improv.		Not shared		6.4
77	MASC	Cognitive	Multi-agent/ Homogenous	Single-role	Virtual ecosystem	–	–	Symbolic	Mess.	C	Rhythm Gen.	✓	Not shared	✓	6.4
78	MASOM	Cognitive	Multi-agent/ Homogenous	Single-role	Real-world	Hybrid	Audio	Audio	Env.	P+L	Improv., Comp.		Partially	✓	6.4

Musical Agents, we focus on purely computational software agents, and exclude the virtual agent applications in CGI, and robotics.

More specifically, the survey covers 78 musical agent systems compiled in Table 1. Certainly, many more musical agents have been developed within the artistic practices. However, we only cover the systems whose details are given in peer-reviewed publications. The systems are referenced throughout the paper using the name convention *system-name* ①, where the circled number refers to the system numbers in Table 1. We propose a taxonomy (Figure 2) that is framed using the terminology of MAS, AI and Computational Creativity (CC). We incorporated established dimensions and categorisations of these fields in our taxonomy rather than coming up with new ones. We aimed for a terminology that is inclusive of both Generative Music (an artistic practice) and Computational Creativity for Music (a scientific research field). In the next section, we supply a background of these associated fields. We present a typology of musical agents, and extend the agent classification of MAS to include the particularities of musical agents and introduce the various dimensions of musical agents in Section 3. We subsequently group musical agents according to their MAS architecture, and present details on each system in Sections 4, 5 and 6. Then, we discuss the evaluation of musical agents in Section 7. In the last section, we propose Musical Metacreation as a field that combines science (Computational Creativity) and artistic practice of Generative Music; and propose possible future directions in the field.

2. Generative Art and Computational Creativity

We build our review using the terminologies of Generative Art, an artistic practice, and Computational Creativity, a scientific field. Before we start the survey of musical agents, we would like to introduce the fields that encompass musical agents. We first make a note of two generic fields, Generative Art and Computational Creativity, then we continue to more specific fields that are Metacreation and Musical Metacreation.

The roots of Computational Creativity can be traced back to Generative Art as well as AI, Artificial Life (A-Life), Machine Learning and Cognitive Sciences. Galanter (2003) defines Generative Art as follows:

> Generative Art refers to any art practice where the artist uses a system, such as a set of natural language rules, a computer programme, a machine, or other procedural invention, which is set into motion with some degree of autonomy contributing to or resulting in a completed work of art.

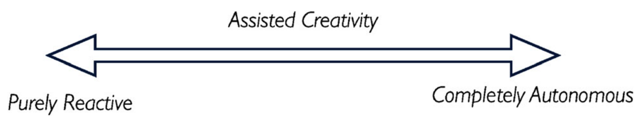

Figure 1. The continuum of autonomy.

We observe in this review that some musical agent systems inherit rules that are strictly defined by their authors whereas other systems adapt their aesthetics by a learning process. That is, the degree of genericity varies in autonomous systems as well as musical agents. The genericity of musical agents thus spans a continuous dimension that ranges from specific systems to purely generic systems. Many rule-based musical agents lean towards to the specific end of the genericity continuum. For example, *Voyager* ⑪ includes 15 pitch generation algorithms that are strictly defined by its creator, George Lewis (2000), as such strictly implements the aesthetics of its creator. In comparison, the *Continuator* (50) can learn the style of any musician and does not include pre-defined music rules. The Voyager is closer to the specific end of this continuum whereas the Continuator stands closer to the generic end.

The notion of autonomy frequently emerges when we discuss generative systems. As Galanter (2003)'s definition emphasises, all generative systems posses a degree of autonomy. Hence, we define a dimension of autonomy that is continuous, ranging from purely reactive systems without autonomy to completely autonomous systems (Figure 1). For example, MuseScore[3] is a music notation software. MuseScore only produces music as a direct result of the user's input and is purely reactive. In contrast, the *Continuator* (50) autonomously learns from its user's input and continues a melody when the user/musician stops playing. The computer assisted creativity would fall in the middle range of the autonomy continuum. An example of computer assisted creativity is assisted composition in music (see Section 3).

Autonomous systems of Generative Art focus on artistic creative tasks. Note that, there are also creative tasks that are not artistic. For example, creating a culinary recipe (Ámorim, Góes, da Silva, & Franćsa, et al., 2017) is a creative task that is not artistic. The academic field, Computational Creativity studies computational processes for all creative tasks including the artistic ones. Creative tasks of art and music are different than problems with optimal solutions. In the case of problems with optimal solutions, the quality measures are well-defined.

[3] https://musescore.org/

For example, we evaluate the performance of a software agent that aims to optimise fuel consumption, by measuring the actual fuel usage. In contrast, creative tasks of art and music tackle problems that lack definitive or optimal solutions. The solutions of creative tasks of art and music have ill-defined quality measures, e.g. there is no notion of optimal music, nor universal measure of quality in musical improvisation. Colton Wiggins (2012) define Computational Creativity as,

> The philosophy, science and engineering of computational systems which, by taking on particular responsibilities, exhibit behaviours that unbiased observers would deem to be creative.

The research in Computational Creativity mainly centres around the following three common themes:

- Computational models of (human) creativity: Such studies research creativity using computational models. In the case of artificial agents, the possibilities can also go beyond human capabilities. For example, Collins (2017) proposes the notion of a musical agent that listens to more music than humans could.
- Computational systems for supporting creativity: These systems are smart assistants for creative applications. These assistants can suggest solutions and alternatives to the user by analysing the user's behaviour. Musical agents focusing on assisted composition are examples of such systems.
- Artificial creative systems: Computational models of creativity are studied through the development of artificial creative systems. We propose to define these systems as Metacreations.

We call Metacreation the domain that both study and produce systems that partially or completely automate creative tasks. The notion of Metacreation and Meta-level creativity was mentioned by many (Boden, 2009; Buchanan, 2001; Wiggins, 2006a,b) and the term was explicitly proposed by Whitelaw (2004). The term also resonates back to the artistic statements (by artists such as Nicholas Schöffer and James Seawright) in the 1950s and 1960s (Whitelaw, 2004).

There are two types of creativity that Metacreation explores. First, the simulation of human creativity is *creativity as it is*. For instance, *Continuator* (50) is a musical agent that implements *musical creativity as it is*, by imitating the musical style of a performer. Second, exploring creative processes that humans are incapable of, is *creativity as it could be*. For example, *Shoals* (44) explores *musical creativity as it could be*, by sonifying the actions of a virtual ant colony.

Building on this literature, Pasquier, Eigenfeldt, Bown, and Dubnov (2017) define *Musical Metacreation* as '… a subfield of Computational Creativity that addresses music-related creative tasks'. We revisit this definition and we propose that *Musical Metacreation* is the partial or complete automation of musical tasks. MuMe, as an interdisciplinary field, is inclusive of all approaches, studies, domains and practices that automatise musical tasks. We acknowledge that several other domains also study the topics of MuMe, and we elaborate on this in Section 8.4 We propose to define MuMe as a field that uses the terminology of Generative Art (practice) and Computational Creativity (science) to cover autonomous systems of algorithmic music, generative music, machine musicianship and machine improvisation. The applications of MuMe use techniques of computational models, Artificial Intelligence (AI) and MAS to automatise musical tasks. Musical agents is a sub-category of MuMe, and in the next section, we delve into musical agents and the musical tasks that are carried out by musical agents.

3. Typology of musical agents

A musical agent is an artificial agent that partially or completely automates musical creative tasks. In the following, we explain the terms 'musical' and 'agent'. We refer to the term musical in the context of Varése's *Organised Sound* (Varese & Wen-chung, 1966). In this survey, the definition of music is inclusive of all works that use sound as a medium. Hence, we also include implementations of Sound Art, Sonic Arts and Contemporary Art works using the sound medium.

Although there is no consensus on the definition of agents in Social Sciences and Philosophy (Emirbayer & Mische, 1998), an agent is a well-defined term in Computer Sciences. An agent is an autonomous system that initiates actions to respond to its environment in timely fashion (Wooldridge 2009). Similarly, musical agents explore the notions of autonomy, reactivity, proactivity, adaptability, coordination and emergence. In this survey, we include musical agents that implement communication but do not implement machine listening. However, we exclude musical agents that solely analyse music, and purely generative systems[4] that have neither perception capabilities nor communication abilities. Some musical agents work offline such as *MASC* (77) while others work

[4] Herremans, Chuan, and Chew (2017) recently surveyed purely generative systems from a conventional music perspective.

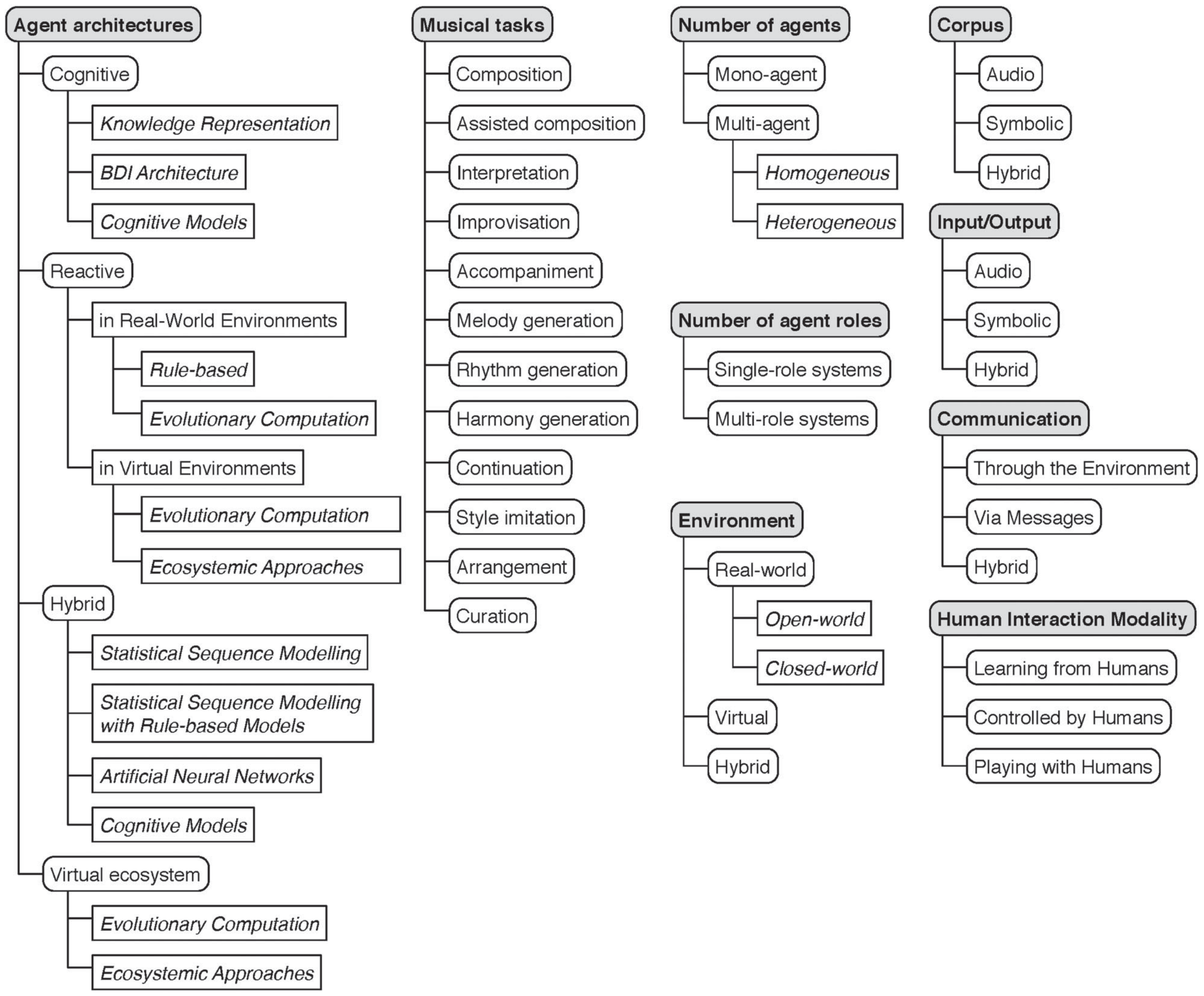

Figure 2. The nine dimensions of our musical agents typology.

online such as the *Contuniator* (50) and the *Voyager* (11). There are also musical agents that learn offline and generate online such as *MASOM* (78).

There is a wide variety of musical agents. We reviewed 78 systems and identified 9 dimensions that form the typology of musical agents. These nine dimensions are agent architectures, musical tasks, environment types, number of agents, number of agent roles, communication types, corpus types, input/output (I/O) types, human interaction modality (HIM). This typology is available in Figure 2 and Table 1.

(1) **Agent architectures:** Our typology of musical agent architectures (Figure 2) is based on well-known agent classifications in MAS and Artificial Intelligence literature. On the top level of the musical agent architecture typology, we classify musical agent architectures using three broad types of agent architectures: cognitive, reactive, and hybrid (Russell Norvig, 2010; Weiss, 2013; Wooldridge, 2009). Under the agent types, we use architecture model paradigms as another level of categorisation. This classification of agent architectures and model paradigms also serves as the base along which we discuss our survey of musical agents, and the details on each agent architecture type are given in the corresponding sections.

(2) **Musical tasks:** Musical agents partially or completely automatise musical creative tasks. So far, we identified 12 different musical tasks implemented by musical agents (Figure 2):
   - **Composition:** The artefacts of composition are sets of symbolic instructions in the case of musical scores, or audio files in the case of fixed-media works in electroacoustic music or acousmatic music. For example, *Coming*

Together:Freesound [25] is a system that generates soundscape compositions.

- **Assisted composition** systems recommend musical ideas to composers by automatising any sub-tasks of musical composition. For example, MASC [60] implements Affective Computing with a MAS to recommend melodies to composers. Also, several composers used *OMAX* [54] to recombine and transform musical material for composition tasks.[5] .
- **Interpretation:** Performers interpret a set of musical instructions to produce sounds or generate audio, which we refer to as interpretation tasks. For example, *IMAP* [36] evolves different interpretations of the same musical phrase using a MAS. Interpretation tasks can also appear in the musical tasks of symbolic (notated) music.
- **Improvisation:** We can break down the improvisation task into real-time distributed composition and real-time interpretation tasks. For example, *MASOM* [78] performs free improvisation with or without software and human agents in the context of experimental electronic music. Moreover, musical agent improvisation is also referred to as machine improvisation.
- **Accompaniment** tasks incorporate following and supporting a leading performer or musical part. For example, *Virtualband* [15] follows the eventfulness of a performer's audio and generates rhythm, chord progressions and bass parts. Accompaniment task can appear in composition, interpretation and improvisation tasks.
- **Melody, rhythm and harmony generation** tasks appear as sub-tasks of composition, assisted composition, interpretation and improvisation.
- **Continuation** consists of having a musician play or improvise, and the system taking over once the musician stops. For instance, the *Continuator* [50] is a musical agent that carries on a musical phrase played by a human performer in the style of the human performer.
- **Style imitation:** Given a corpus $C = C_1, \ldots, C_n$ representative of style S, style imitation is to generate new instances that would be classified as belonging to S by an unbiased observer (typically a set of human subjects). For example, the *Audio Oracle* [58] is a musical agent that uses machine listening to imitate the style of another performer.
- **Arrangement:** The selection and temporal ordering of musical material are the main tasks of musical arrangement. We differ arrangement from instrumentation which is the assignment of parts of the music to specific musical instruments. In the case of musical agents, we encountered only one system, the fourth version of *Coming Together* [25] that implements arrangement.
- **Curation** differs from arrangement. Curation is the selection of agents to perform whereas arrangement is selecting and ordering musical material. For example, *ParamBOT* [27] is a musical agent that curates a selection of musical agents.

These musical tasks are neither mutually exclusive nor independent. For instance, a composition task may include sub-tasks of melody, rhythm and harmony generation as in the case of *Inmamusys* [2]. In contrast, *Coming Together:Freesound* [24] also implements composition tasks, and yet, it does not include any of the sub-tasks of melody, rhythm and harmony generation.

(2) **Environment types:** The literature considers three types of environments in musical agent systems. First, the *real-world* environment is the sound medium where an agent listens to the sum of all sounds generated by all agents. For example, in a duo setting, *Voyager* [11] listens to the human performer and outputs audio to the real-world environment so that the human performer can hear. There are three types of agents in real-world environments: physical, visual and sonic. Physical agents in real-worlds are musical robots, and visual agents apply visualisation of artificial agents. We mentioned that we do not cover robotic and CGI agents in this survey and we only cover sonic software agents that listen to either audio or symbolic music input. Furthermore, there are two sub-categories of real-world environments: *open-world* and *close-world*. Open-world environments allow human or software agents to enter and leave the environment during the generation stage whereas close-world environments do not. For example, *Voyager* [11], *Odessa* [16] and *MASOM* [78] listen to the real-world and allow human or software agents to enter and leave the environment.

Second, simulations of physical environments are *virtual environments*. In the literature, musical agent architectures utilise a virtual environment in three ways to generate audio: the virtual location of an agent, the spatial interactions between agents in the

[5] Musical examples of OMAX in practice can be found at http://repmus.ircam.fr/omax/home

virtual space, and an agent's interaction with the virtual environment such as finding virtual foods. For instance, the agents in *Shoals*' [44] are situated in a virtual environment where they consume virtual foods. The system generates audio by sonifying the consumption of food. Also, the agents can create groups by communication through spatial interactions in Shoals. Third, the real world environment affects the virtual ecosystem in hybrid environments. For example, the video input in *Petri* [38] generates attraction points in the virtual environment. The agents try to move towards these attraction points in the virtual environment and the location of agents create the audio output. Moreover, the following properties of MAS environments are also applicable to musical agent environments (Russell Norvig, 2010):

- *Fully observable vs. partially observable:* An environment is fully observable if an agent can perceive the environmental properties that are relevant to the choice of action. An environment is partially observable if an agent has no capabilities to perceive. For example, an agent in the system [35] perceives only one agent at a time although there are multiple agents in the system.
- *Deterministic vs. stochastic:* An environment is deterministic if the next state of the environment only depends on the previous state of the environment and the actions of the agents in the environment, such as the environment of *Beatbender* [20]. Non-deterministic and partially observable environments are called stochastic environments. There are two types of stochastic environments: stationary and non-stationary. In stationary stochastic environments, there is only one stochastic model and the model does not change. For example, the probability distributions in *Virtualband* [15] does not change during the performance. In non-stationary stochastic environments, the stochastic model changes. The *Continuator* [50] is an example of an agent in non-stationary environment because the agent's stochastic model, that is the Markov model, changes continuously.
- *Episodic vs. sequential:* During each episode, an agent has a percept input and generates an action output. In an episodic environment, the current episode of an environment is independent of the previous episodes. For example, each time *GenJam* [28] starts playing a chorus, the solo is independent of the previous choruses' solo. In a sequential environment, the current episode depends on the previous episodes. For instance, the musical agent applications with the first or higher order Markov Models have sequential environments.
- *Static vs. dynamic:* An environment is static if it does not change while an agent is deliberating, else it is dynamic. An example of dynamic environment is the virtual environment in Petri [38] because the attraction points in the virtual environment change independently.
- *Discrete vs. continuous:* An environment is discrete if it has a finite number of distinctive states, and continuous if the environment has an infinite number of distinctive states. The applications with symbolic music representation have discrete environments when the parameters are discrete, such as pitch values in MIDI. In comparison, an audio environment is continuous.

(3) **The number of agents:** We group musical agent systems in two categories with reference to the number of agents included: *mono-agent* and *multi-agent*. Mono-agent systems include only one musical agent whereas MAS have many. Although we approach human performers as agents, we only include software agents in this categorisation. Also, we present the interactions of musical agent systems with humans in the Human Interaction Modality (HIM) dimension. Human performers can play with a mono-agent system. For example, the *Continuator* [50] is a mono-agent system in a duo setting with a human performer. MAS in which all the agents share the same architecture are said to be a *homogeneous* MAS. For example, *MASOM* [78] has a flexible homogeneous architecture that allows the user to start more than one *MASOM* agent for a live performance. A musical agent system is a *heterogeneous* Multi-agent system if there are agents with different architectures. For example, *Cypher* [10] is a heterogeneous Multi-agent system with multiple agent architectures.

(4) **The number of agent roles:** All agents focus on the same task in *single-role* systems. In comparison, there are different roles that agents can take in *multi-role* musical agent systems. For example, there is only one type of agent architecture in *iME* [48], and agents can take the roles of either *listener* or *player* in an episode. Agents in *iME* can change their roles every episode.

(5) **Communication types:** There are three types of communication in musical agent systems: *through the environment, via messages* and *hybrid*. First, through the environment communication is related to the notion of *stigmergy*. Stigmergy is the indirect coordination through the environment (Heylighen,

2016). That is, an action of an agent leaves footprints in the environment. These footprints stimulate the agents in the environment. For example, ants leave traces when they find food in the environment (Sumpter Beekman, 2003). Other ants follow these traces to reach the food. Similarly, musical agents implement machine listening of the real-world to communicate through the environment. For example, let's assume that a musical agent desires the ensemble to play louder. An agent expresses this desire by playing louder sounds instead of sending symbolic messages to the other agents in the ensemble. The other agents interpret this desire by listening to the real-world environment.

Second, agents that communicate via messages use pre-defined, system specific messages. The developers of musical agents come up with protocols that specify the type of messages, and how agents send and receive messages. For example, the agents in *Inmamusys* $\textcircled{2}$ communicate via messages to generate compositions as notated symbolic music.

Third, both of these methods are used in the *hybrid* communication. For instance, *Indifference Engine* $\textcircled{5}$ communicates through environment by listening to a human performer and generating audio, while the agents in the system communicates through system specific messages such as *pitch, volume, speed, tensionCurve,* and *confidence*.

In MAS, the communication between agents includes negotiation, bargaining and argumentation (Weiss, 2013). An agent's behaviour can be cooperative or competitive. For example, playing chess is a competitive behaviour whereas distributed problem solving is cooperative. Musical agent implementations are mostly cooperative given that the nature of making music is cooperative. Nevertheless, there are also examples of systems with competitive agents. For example, agents create social groups in *Shoals* $\textcircled{44}$, and these groups compete with each other to find and consume food in a virtual ecosystem.

(6) **Corpus types:** A corpus is a set of symbolic music or audio samples in the case of musical agents. Musical agents use a corpus as a musical memory and knowledge. We group the corpus types of musical agents in three categories: symbolic, audio and hybrid. A symbolic corpus is a set of symbolic representations of music. In most of the cases, the symbolic representation uses MIDI. An audio corpus is a set of audio samples. The samples in an audio corpus can range from grains of audio samples to full length music pieces. A hybrid corpus include a set of audio files with any kind of symbolic data. For example, *MASOM* $\textcircled{78}$ includes a hybrid corpus, that is, a set of audio segments with 18 dimensional feature vectors.

(7) **Input/Output types:** We differentiate corpus types from I/O types. Corpus types are related to the learning and generation. However, the I/O types clarify how an agent listens and outputs to the environment. We observed three types of I/O in musical agents: *audio, symbolic*, or *hybrid*. The audio I/O is the audio signal perceived and generated by the agents. The agents with the symbolic I/O use a symbolic representation of music, that is, in most cases the MIDI protocol. Agents with the hybrid I/O use both audio and symbolic I/O.

(8) **HIM:** Three types of Human Interaction Modality (HIM) emerge in musical agents: *systems learning from humans, systems controlled by humans,* and *systems playing with humans*. Some musical agent systems learn the style of a human performer or composer as in the case of the *Continuator* $\textcircled{50}$ and *MASOM* $\textcircled{78}$. Some systems include global variables that can be controlled by humans as in the case of *Kinectic Engine* $\textcircled{21}$ and *HARP* $\textcircled{71}$. Many of the musical agents, such as the systems focusing on machine improvisation in Section 6.1, can perform with human performers.

In the following sections, we define the agent architectures, group musical agents by their architecture type, and give details about each system.

4. Cognitive musical agents

Gomila Müller (2012) define a cognitive system as one that 'learns from individual experience and uses its knowledge in a flexible manner to achieve its goals '. Six aspects of cognitive systems are dealing with an uncertain world, learning from experience, understanding knowledge, flexible use of knowledge, autonomy and social abilities. Cognitive agents inherit the properties of both cognitive systems and MAS. Cognitive models, in comparison, are software architectures that specifically model human cognitive processes. Some examples of cognitive models proposed in Cognitive Science are Ymir, ACT-R, Soar, NARS, OSCAR, AKIRA, CLARION, LIDA and Ikon Flux (Thórisson Helgasson, 2012). We study cognitive musical agents in three categories in the following sections.

4.1. *Cognitive musical agents with knowledge representation*

A classic paradigm for cognitive agent architectures is *Logic-based agents* (LBAs) (Weiss, 2013; Wooldridge, 2009).

LBAs perceive their environment by building and maintaining symbolic representations. The environment is represented as a set of assertions. LBAs reason about the environment using a set of logical rules and apply theorem-proving using the knowledge.

A recurrent theme in cognitive musical agents is the implementation of knowledge representation with a rule-based agent architecture. Inspired by music theory, the authors come up with a knowledge representation, and rules of their musical system. Alternatively, the agents generate logical assertions and rules using a corpus. In the following, we survey three systems with knowledge representations.

Wulfhorst, Nakayama, and Vicari (2003) studied 50 popular songs to devise a table of possible harmonic transitions. Vicari, Nakayama, Wulfhorst, Costalonga, and Miletto (2005) continue this work by presenting a multi-agent system with multiple agent models, called Virtual Musical Multi-agent system, *VMMAS* ①. The application of this system is online music accompaniment. Seven types of agent models appear in *VMMAS* ①: Cautious (activates only when the agent has a high metric and harmonic confidence degree), Leader (simulates other agents), Flexible (adapts to the metric changes), Inflexible (does not adapt to the metric changes), Persuasive (tries to stabilise around its 'ideal' tempo), Improvising (proposes harmony progressions) and Lyric (proposes tempo changes). The authors state that agent models use 'fuzzy cognition'; however, the authors have concealed how the fuzzy logic is implemented.

The second system, *Inmamusys* ② concentrates on a two-layer multi-agent system (Delgado, Fajardo, & Molina-Solana, 2009). All agents in *Inmamusys* include a knowledge representation. In the first layer, Inmamusys ② chooses a composer agent that decides the number of voices, the timbre of the voices, tonality and the number of measures. In the second layer, there are four types of agent models: melody, harmony, accompaniment and drums. The system allows a degree of human control through a graphical user interface (GUI). In this interface, the user can choose the desired emotion, instruments and duration of the composition. However, it is not disclosed how desired emotions are implemented.

The third system is a generative multimedia system ③ with affective computing (Bizzocchi et al., 2015). This system generates soundscape, moving images and music. The affect estimation uses Russell (1980)'s two dimensional (valence and arousal) circumplex affect model. The system design includes three modules: Re:Cycle, AuMe (Audio Metaphor) and Musebots (see Figure 3). The Re:Cycle uses a corpus of moving images with valence and arousal tags. Re:Cycle module sends the desired valence and arousal values to AuMe and Musebots. Using a machine learning model (multivariate regression) for affect estimation in soundscape recordings, AuMe chooses soundscape recordings with the desired valence and arousal values. The Musebots module maps valence (pleasantness) and arousal (eventfulness) to multiple musical parameters such as musical consonance, melodic movements and rhythm.

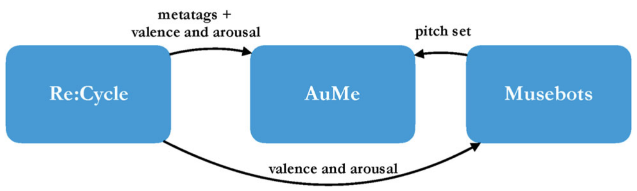

Figure 3. The block diagram of the system ③ in Table 1 (Bizzocchi, Eigenfeldt, & Thorogood, 2015).

4.2. *Cognitive musical agents with BDI architecture*

One of the most common cognitive agent architectures is the Belief-Desire-Intention (BDI) architecture (Wooldridge, 2009). The BDI architecture[6] applies *practical reasoning*. Practical reasoning is the goal-directed selection of actions. There are two aspects of practical reasoning: *deliberation* and *means-end reasoning*. Deliberation is the process of deciding what to achieve. The outputs of deliberation are intentions. Means-ends reasoning is how to achieve the goal that is set by the deliberation process. The result of means-ends reasoning is a *plan*.

An agent uses its beliefs to represent the state of the world and the know-how of the agent. Beliefs are different from the knowledge. Beliefs can be wrong and they are non monotonic. Internal or external perception update beliefs. Internal perception is the perception of the agent's own state whereas the external perception is the perception of the environment. Reasoning can also update the beliefs of an agent. There are two families of reasoning: perfect and bounded rationality. Perfect rationality is where we assume the logical omniscience of the agent. In comparison, bounded rationality is the notion of accepting the finite nature of the resources available for reasoning.

While beliefs are informational attitudes, the desires are motivational attitudes. Desires are not necessarily consistent or achievable. Deliberation is the process of choosing which desires are to be pursued according to the current beliefs. The agent generates intentions by applying a selection function to its desires.

[6] Wooldridge (2009) provides the psuedo code of the BDI agent control loop.

Intentions are desires that the agent is committed to make happen. Agents determine ways to change the state of the environment so as to make the intentions true. Intentions provide a filter for the adaptation of the other intentions that must not conflict. Agents track the success of their intentions. Agents are inclined to try again if the attempts to satisfy an intention fail. That is, the intentions of agents are persistent. Agents believe that their intentions are possible and intentions are not closed under implications.

Means-end reasoning is to determine how the intentions are achieved. Agents generate a sequence of actions, that is, a plan. Plans can be deterministic and non-deterministic. Means-end reasoning generates or selects a plan to be executed as an attempt to achieve the intention.

We found three musical agent implementations that use BDI architecture explicitly. The first system ④, presented as a part of the Coming Together musical agent series, explores the idea of negotiation between musical agents (Eigenfeldt, 2010). The communication is hybrid and through communication, the agents generate their own goals and create plans to achieve these goals. An agent in this system desires to create a repeating phrase that is updated by the communication between agents and the changes in the environment.

Indifference Engine ⑤ is the second musical agent application including BDI architecture with the hybrid communication (Eigenfeldt, 2014). Each agent generates an intention graph for *pitch*, *volume* and *speed* at the beginning of performance. The average of these three graphs is the *tensionCurve* of an agent. Indifference Engine can run as a mono-agent system, multi-agent system and multi-agent system including a human performer. The data of agents are globally shared; however, the intentions of agents are private. The agents also negotiate their intentions and argue on choosing a leader. Each agent follows the leader with a weight parameter called *confidence*. The confidence parameter is set by the proximity of the agent to the mean of other agents' pitch, volume and speed parameters. The closer an agent is to the mean, the higher its confidence is. Moreover, the agents choose to either join the multi-agent ecosystem or follow the human performer. Also, the agents negotiate on which audio corpus to use at the beginning of performance. The musical agents in this system generate sounds with concatenative synthesis using CataRT.

The third system with the BDI architecture is *MUSIC-MAS* ⑥ (Navarro, Corchado, & Demazeau, 2014; Navarro et al., 2016). The focus of this study was assisted composition and style imitation. *MUSIC-MAS* generates harmony progressions using organisation-oriented MAS design to introduce flexibility and scalability to the system design. *MUSIC-MAS* implements a client-server based MAS architecture proposed by Ferber, Gutknecht, and Michel (2003) including multiple agent types (Figure 4). *ProviderAgent* assigns certain roles to *ClientAgents*. *MUSIC-MAS* has five client agent roles: *composer, evaluator, interface, data supplier* and *control*. Composer agents implement BDI architecture to generate harmony progressions. Evaluator agents score the generated progressions using a fitness function that includes harmony progression constraints. These constraints are style specific and the authors state that the system is flexible enough to imitate any style by changing the set of constraints in the fitness function. The interface agents handle the interaction with the user. The data supplier agents collect and store all the data generated by the system. The control agents are responsible for the communication and coordination of the agents.

4.3. Cognitive musical agents with cognitive models

Maxwell et al. (2009) propose the framework called *Hierarchical Sequential Memory of Music (HSMM)* ⑦. *HSMM* is build upon the *Hierarchical Temporal Memory (HTM)* (George, 2008). The MuMe applications of HTM are recognition, generation and continuation. Following *HTM*, Maxwell, Eigenfeldt, Pasquier, and Gonzalez Thomas (2012) present *MusiCOG* ⑧, a cognitive musical agent generating monophonic melody. MusiCOG is a mono agent system that applies the MuMe tasks of continuation and assisted composition. *MusiCOG* ⑧ listens to and learns from a sequence of MIDI data. *MusiCOG* consists of four modules: *perception module, working memory, cueing model* and *production module* (Figure 5). The perception module implements segmentation as well as the generation of monophonic MIDI streams from a polyphonic input. Working memory is the short term memory that implements grouping of similar patterns. The cueing model is *MusiCOG*'s long term memory which learns the hierarchical structures between patterns in the

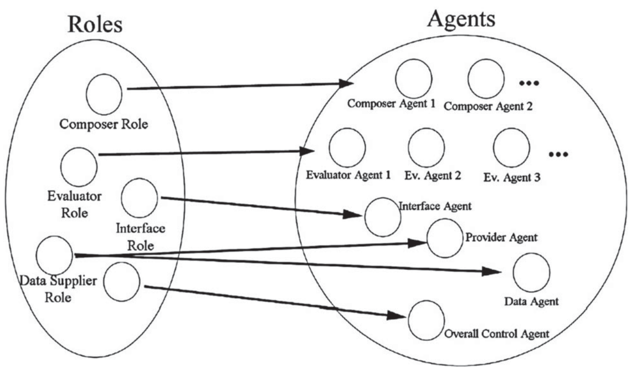

Figure 4. An example of role assessment in MUSIC-MAS (Navarro, Corchado, & Demazeau, 2016).

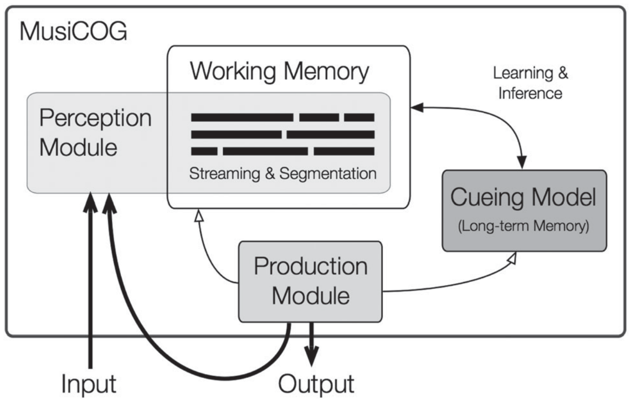

Figure 5. The architecture of MusiCOG (Maxwell, Pasquier, & Eigenfeldt, 2009).

working memory. Finally, the production module generates monophonic MIDI output using the knowledge representation in *MusiCOG*.

In regards to communication between musical agents, Murray-Rust and Smaill (2005), Murray-Rust, Smaill, and Edwards (2006), Murray-Rust (2008), Murray-Rust and Smaill (2011) proposed the Musical Acts theory ⑨ that was inspired by the Speech Acts theory (Kimball, 1975). Murray-Rust and Smaill (2011) propose three qualities of Musical Acts:

- *Embodiment* through the production of music, . . . musical acts must have a manifestation in music.
- *Intention* is what differentiates a musical act from general musical playing. A musical act should have perlocutionary force.
- *Intelligibility* is necessary for a successful act; if it is not understood, then it will fail to change the world, as other musicians will fail to react to it.

The information in Musical Acts is generated by *descriptors*. A descriptor is a mapping from a *facet* (an aspect of music, such as melodic contour, chord, time signature) to a *value*. An *analyser* generates a descriptor in Musical Acts Theory. The theory also includes a set of performative actions (*inform, confirm, disconfirm, extend,* and *alter*) to create a dialogue between musical agents. Murray-Rust and Smaill (2011) also present a MAS with Musical Acts using a symbolic representation of music. The agents in the MAS were trained on the piece Canto Ostinato by Simeon ten Holt. The agents analysed levels, slopes and patterns for note timing, length and loudness features using the symbolic music.

5. Reactive musical agents

Reactive agents respond to the changes in the environment without the explicit symbolic reasoning of the type carried out by cognitive agents. In MAS, there are two types of reactive agents architectures: *reflex* and *reactive*. Reflex agents do not have any internal states. The perceived states of the environment, *percepts* cause actions of reflex agents. Hence, the simplest agent architecture is a reflex system, that is a function that maps percepts to actions:

$$f : P \to A,$$

where f is a function, P is a set of percepts and A is a set of actions. Unlike reflex agents, reactive agents have internal states. These internal states are functional as opposed to cognitive.

A well-known reactive agent architecture is the Subsumption architecture (Brooks, 1986). The Subsumption architecture is hierarchical with multiple layers. Higher layers have a higher priority and vice versa. Outputs of higher layers can restrain, alter, or block outputs of lower layers.

Moreover, Brooks (1995) discusses four key terms of the Artificial Intelligence and MAS research: *Situatedness, Embodiment, Intelligence* and *Emergence*. Situatedness proposes that the intelligence is situated in the interaction with the environment, responding to percepts in a timely fashion, rather than reasoning about the environment through a symbolic representation. Embodiment is the idea that an agent is an 'embodied intelligent agent' and intelligence is situated in the real world through physical grounding. Brooks (1995) claims that the interaction between an agent and its environment is the determinant of the intelligence of an agent. Therefore, intelligence emerges as a result of the interaction between the behavioural rules of the agent and its environment.

In the following, we survey 39 systems with reactive musical agents. We group these systems according to their environment types, reactive musical agents in real-world and virtual environments. Reactive musical agents in real-world environments appear in two categories: rule-based reactive musical agents and agents with Evolutionary Computation (EC). In the reactive musical agents in virtual environments, musical MAS simulate virtual environments to conduct a musical task. We group the reactive musical agents in virtual environments into two categories of multi-agent simulations with EC and multi-agent simulations with ecosystemic approaches. We differentiate the reactive musical agents that use EC to generate musical material from the musical MAS that use EC to evolve agents. The systems that we cover in Section 5.1.2 implement EC to generate musical material within a reactive agent architecture in real-world environments. We cover Multi-agent simulations that utilise EC to evolve agents in virtual environments in Section 5.2.1.

5.1. Reactive musical agents in real-world environments

We group musical agents in this category in two: rule-based reactive musical agents, and reactive musical agents with Evolutionary Computation.

5.1.1. Rule-based reactive musical agents

A recurrent theme in reactive musical agents is the application of music theory rules in the design of musical agents. Rule-based systems apply percept-to-action functions as *IF-THEN* conditionals. In most cases, these rules are strictly set by the designer of the system. We start our survey of rule-based reactive musical agents with the systems automatising the improvisation tasks.

One of the early musical agent is *Cypher* ⑩, a rule-based reactive musical agent working with symbolic representation of music (Rowe, 1992). Cypher is a multi-role, heterogeneous Multi-agent system. Cypher is also an example of *holonic* Multi-agent systems. In holonic systems, an agent is made of other agents (Minsky, 1986). In Cypher, the agents are hierarchically arranged and connected. At the highest level, there are two types of agents, listener and player agents. First, listener agents behave similar to the perception modules, analysing the input data and providing high-level musical information to player agents. The listener agent is made of the register agent, the dynamic agent, the density agent, the speed agent, the duration agent and the harmony agent. Each of these agents implement a particular Music Information Retrieval (MIR) task. Second, player agents generate musical output in three ways: transformation of the input data (symbolic representation of music), triggering events by using the information provided by the listener agent, and choosing from a corpus of musical events. Rowe (1992) also mentions that the listener agents can be used as critics that analyse the output generated by player agents. Listener agents as critics report the success of previous events to player agents. This application of critic agents is similar to the reinforcement learning in statistical sequence modelling algorithms (see Section 6.1).

Voyager ⑪ is one of the well known musical agents (Lewis, 2000). Lewis mentions that the first version of Voyager goes back to 1986. The global behaviour mode of Voyager determines timbre, volume range, microtonal transposition, tempo, tactus (underlying, inner pulse), note probability distributions, pitch interval range, inter-onset time intervals, pitch set and pitch generation algorithm. Voyager has 64 MIDI-controlled, monophonic *player* agents. Depending on the global behaviour mode, Voyager groups or separates these 64 agents. Voyager changes the global behaviour mode every 5–7 s. The system includes 15 pitch generation algorithms and 150 microtonal pitch sets. Moreover, *setresponse* module handles the responses to the input data by modifying the parameters set by the *setphrasebehaviour* module. As of 2017, Voyager still performs in various venues. As a part of the Musical Metacreation concert at ISEA2015 in Vancouver, BC, Canada, Voyager improvised with two human performers playing prepared piano and clarinet.

Band-out-of-a-Box, Bob ⑫ is another one of the early reactive musical agents. Bob improvises with a human performer in the context of jazz and blues (Thom, 2000a, b, 2003). Bob focuses on the generation of melodies, specifically *solo trading* in jazz improvisation. In solo trading, musicians improvise one by one for a number of measures. The number of measures is an integer division of the total number of measures in a chorus. In the jazz context, it is common to trade solos in four or eight measures. Bob improvises with a human performer, along with a fixed musical accompaniment (Figure 6). Bob utilises trees to represent a measure, and implements histograms of the pitch class, intervals and melody direction to learn the playing modes of the human performer. The real-time generation of musical sequences is a mapping that associates the histogram of feature vectors to the playing modes of Bob. Bob generates these playing modes during the learning, and stores them as a knowledge representation. However, these playing modes do not provide a temporal pitch sequence. To generate a melody, Bob implements a first-order Markov chain in which the states consist of an absolute pitch value and a set of histograms. Thom (2003) also demonstrates Bob's performance by training two different agents on Stéphane Grappelli and Charlie Parker solos.

Adaptive Real-time Hierarchical Self-monitoring (ARHS) ⑬ emphasises timbre in the reactive musical agents (Hsu, 2010). The system includes three

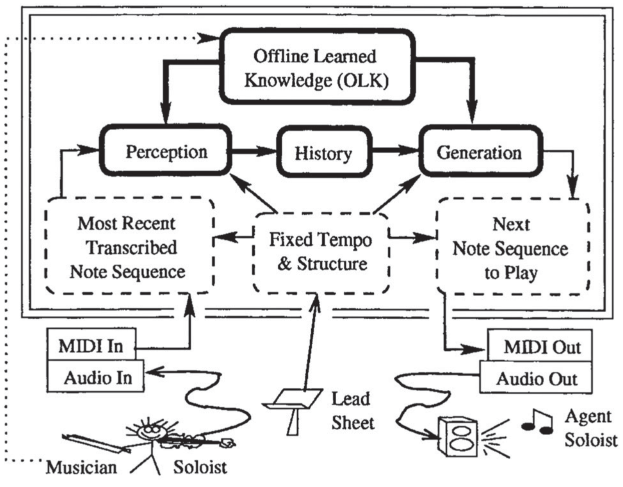

Figure 6. Bob's system architecture Thom (2003).

modules that are *Sensing, Synthesis* and *Processing*. The three module architecture of AHRS resembles Blackwell et al. (2012) *PQf* musical agent framework in which *P* is the machine listening and analysis; *Q* is the synthesis and *f* implements reasoning and generative functions. We further discuss the *PQf* approach in Section 8.4. The *sensing* module of AHRS includes pitch, amplitude, loudness, tempo, auditory roughness and timbre analysis. The sensing module outputs a performance mode that is updated every second. Then, the processing module uses the performance mode to choose sounds. There are two features in the processing module: short-term responsiveness and long-term adaptivity. The system initiates short-term responses using the Sensing module output and fuzzy logic. Short-term responses are beginning a phrase, ending a phrase and changing the timbre. The long-term adaptivity implements fuzzy sound selection based on 8-second windows of the Sensing module output. The Synthesis module includes one or more agents. The Sensing and Processing modules set global variables for these agents. Agents generate parameter curves to change pitch and timbre characteristics.

ListeningLearning (LL) ⑭ is a system that performs with humans in the context of free improvisation (Collins, 2011). The listening algorithm has three modules: rhythm tracking, silence detection and timbral state clustering (Figure 7). The rhythm tracking module calculates onset detection, periodicity and inter onset interval. Also, the rhythm tracking module differentiates free time playing from steady metric tempo. The silence detection module utilises perceptual loudness. LL clusters timbral states using the audio feature statistics of cosine basis energy, cosine-wise inter-frame flux, RMS amplitude, spectral centroid, spectral irregularity and spectral energy. The author normalises six features of timbral state clustering using the *adaptive distribution model*. The adaptive distribution model uses the statistics of feature vectors to implement a normalisation that is similar to histogram equalisation in Computer Vision. LL implements the timbral state clustering using online k-means clustering with the Euclidian distance metric to follow the timbral choices of the human performer. LL includes ten agents to generate output. Each agent corresponds to 1 of 10 timbral states. Only one agent is activated at a time depending on the timbral state calculated by the machine listening module. Each agent has a unique set of parameters for audio synthesis and processing modules. Each agent has 4 audio synthesis and processing modules: sample based drum kits, a physical modelling synthesis of the vocal tract, a 4-voice subtractive synthesis and 50 different audio effects. Collins (2011) states that using ten agents provides the system the diversity of responses required for free improvisation with human performers.

Virtualband ⑮ is a MAS in which musical agents imitate playing styles of musicians (Moreira, Roy, & Pachet, 2013). An agent learns from recordings of a musician to imitate the style of the musician. Each agent is also provided with a lead sheet including the chord progression during learning. Hence, agents are also aware of the harmonic content. Two types of agents appear in Virtualband: *master* and *slave*. Slave agents follow the master agent's eventfulness. The master agent can be a human performer or a software agent. The slave agents have a hybrid corpus including audio excerpts and the audio feature distributions of these excerpts. The duration of excerpts is set to one beat or one bar before the learning. In the generation mode of Virtualband, a slave agent generates output by using concatenative synthesis and chooses audio files to play by following the eventfulness of the master agent. Slave agents conduct the follow role using two mappings: the mapping between audio feature distribution of the master agent and the audio feature distribution of a slave agent, that is, *feature-to-feature (f2f)* mapping. *f2f* uses percentile function to map between two feature distributions. The authors also present case examples of jazz improvisation, interactive mash-ups and beatboxing with Virtualband. The advantage of Virtualband is that the symbolic corpus is not necessarily tied with the audio corpus. That is, we can use the same feature distributions with different audio corpora. Examining the example performances of Virtualband, we observe that the algorithm is successful on jazz improvisation. However, it is not clear why the authors choose the percentile based *f2f* mapping. In the examples of Virtualband, agents are trained on a small corpus, performances of one or two songs.

Odessa ⑯ is a reactive musical agent based on the Subsumption architecture (Linson, Dobbyn, Lewis, & Laney, 2015). Odessa is a mono-agent system that plays improvised music using audio input and MIDI output. Arranged from the lowest to the highest layer, there are three layers in the architecture: *play, adapt* and *diverge*. The *play* layer generates the MIDI output. The *adapt* layer alters the output of the *play* layer according to the input when Odessa listens to other performers using machine listening algorithms for pitch and loudness estimation in the *adapt* layer. The diverge layer constructs higher level musical section changes.

Following the systems focusing on the improvisation tasks, we present five systems generating rhythm: ⑰, *VirtuaLatin* ⑮, *DrumTrack* ⑲, *BBCut2* ⑳, *Kinectic Engine* ㉑ and *BeatBender* ㉒. Pachet (2000) presents a rule-based reactive agent ⑰ that generates rhythm

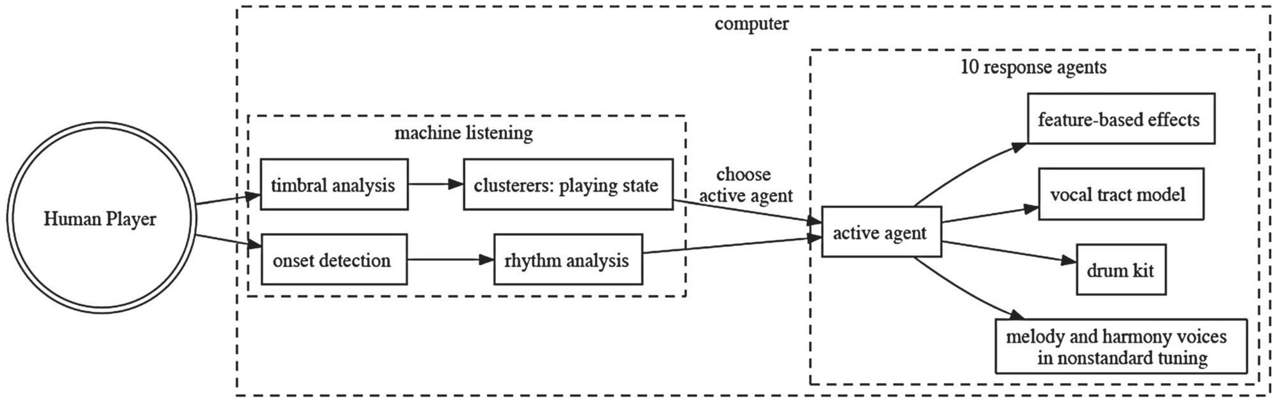

Figure 7. The architecture of LL (Collins, 2011).

and harmony progressions. The agents implement two sets of rules to produce rhythms. The first set of rules create rhythms from scratch using three rules: *emphasise strong beat*, *emphasise weak beat* and *syncopation*. The second set of rules generate variations of existing rhythms with three rules: *add random, remove random* and *move pitch*. To generate harmony, the agent applies the same rules in the vertical dimension (pitch) as opposed to the horizontal dimension (time). Also, the agents apply two additional rules (called *attraction* and *repulsion*) in the horizontal dimension (time) to handle harmony progressions.

Murray-Rust et al. (2005) present another rule-based, rhythm generating MAS. *VirtuaLatin* (18) generates *timbales* accompaniment to Salsa music that is pre-recorded as MIDI files. *VirtuaLatin* works offline. The perception module extracts higher level information from input data on four dimensions: activity, harmonic information, rhythmic information and musical section. The agents select one rhythm from a corpus using the symbolic representation of the song. The agents apply three more rules to introduce ornamentation: *phrasing, chatter* and *fills*.

DrumTrack (19) is another musical agent generating rhythm to accompany a human improvisor (Collins, 2005). DrumTrack focuses more on how to track the tempo of a human performer real-time using machine listening. Although the architecture of DrumTrack is reported as rule-based, the details are not presented.

BBCut2 (20) is the second version of Collins (2006) musical agent generating *breakbeat*. Within our knowledge, the details of the first version are not published. Breakbeat refers to either a musical section where all instruments stop and a drum solo begins,[7] or the

[7] One of the most famous breakbeat sections is the *Amen Break*, which is a 4 bar drum break of the song 'Amen, Brother' by 1960s soul/funk band the Winstons. The recording of Amen Break can be found at http://en.wikipedia.org/wiki/Amen_break.

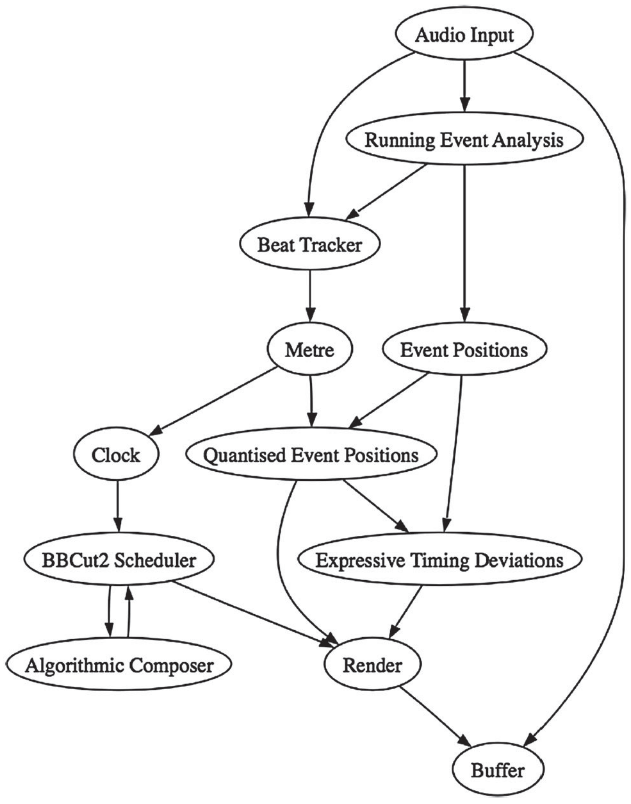

Figure 8. The structure of BBCut2 (Collins, 2006).

electronic music genre in which breakbeat drums are sampled to make mash-ups. BBCut2 uses machine listening techniques to analyse an audio signal and implement beat tracking and segmentation (Figure 8). The agent has an algorithmic composer module. The rules in the algorithmic composer module include generative streams of events, static sequences, shuffling the order sequences, weighted choices, nested patterns and more.

Eigenfeldt (2008) presents the *Kinectic Engine* (21), a rhythm generator with multiple reactive musical agents. The Kinectic Engine includes two types of agents: *conductor* and *player*. There is only one conductor agent, and this agent handles user interaction, identification of player agents, communication between player agents, and sending the tempo to the player agents. Eigenfeldt (2008) implements *personality traits* to vary the player agents' behaviours. Personality traits are presets that determine the density, the amount of rhythmic variation, and timbre of an agent. Player agents decide to play at a certain time using fuzzy logic. The author proposes two types of rules to generate rhythms with player agents: density spread and pattern matching. Moreover, the communication between player agents has two interaction modes: rhythmic polyphony and rhythmic heterophony. The player agent decides on the interaction mode by calculating the rhythmic similarity rating between the agent's own output and the other agents' output. Higher similarity results in the alteration of generated output to satisfy rhythmic heterophony. Likewise, lower similarity alters the agents' generated output towards rhythmic polyphony. The author develops the Kinectic Engine further by introducing Evolutionary Computation to the system (see Section 5.1.2).

BeatBender models a drum circle with the Subsumption architecture to create emergent musical rhythms (Levisohn & Pasquier, 2008). Each rule has an antecedent and a consequent. Antecedents dictate a set of preconditions for a specific rule to be activated. The consequent is the result of an agent's actions when a rule is applied. Each beat, agents decide to play (or not) using their percepts and a set of rules. *BeatBender* includes four types of rules:

- *Collective*: rules that use the total number of active agents
- *Directed*: rules that check states of specific neighbouring agents
- *Temporal*: rules that use the histogram of an agent's states
- *Undirected*: rules that check states of any neighbouring agents.

Following the musical agents applying musical tasks of improvisation and rhythm generation, we continue our survey with the remaining rule-based reactive musical agents. *Andante* ㉓ is a musical agent framework for mobile platforms (Ueda & Kon, 2003). The framework is inspired by the client/server model in MAS. The implementation rests on Aglets Software Development Kit and JAVA Sound API. The authors also present a MAS with agents generating monophonic melodies using different types of noise such as pink, white and brown noises. Although the generation is MIDI, there are built in synthesisers that agents can choose from, to generate the audio output.

Public Space Interactive Web-based Composition System (*PIWeCS* ㉔) is a browser-based multi-agent system that generates musical compositions (Whalley, 2004). The system has an audio corpus of Maori instrument samples. The user can set three variables: *unity/variety, volume and tempo*. The interface includes a conversational model of interaction between the machine and the user. There are four agent types in PIWeCS: *reception, helper, learner and extender*. However, the details of the system architecture as well as the user interface and interaction model are not disclosed.

Eigenfeldt and Pasquier (2011a) present a unique application of generative soundscape composition with MAS including four musical agents, called *Coming Together: Freesound* ㉕. The system is a part of the musical agent series called *Coming Together*. The agents react on the environment, and also communicate using the blackboard architecture to choose sounds to play. The corpus of an agent is labelled by spectral contents and the metadata tags of *voices, animals, water and outside*. There are three types of agent interactions in this system: sharing the metadata tags on a shared blackboard, reacting to the spectral content of the sonic environment, or both.

The fourth version of the *Coming Together* series includes a musical agent ㉕ that concentrates on musical arrangement (Eigenfeldt & Pasquier, 2012). The other agents in the system generate a corpus of musical sections with symbolic music representation (as MIDI files). The arranger agent chooses a random musical section from the corpus, that is, a MIDI file as the first movement. Then, the arranger agent calculates the similarity of the first musical section and the remaining ones on the dimensions of *cumulative density*, *pitch range and variation*, *volume, overall length, specific instrument presence* and *harmonic movement*. The arranger agent sorts the remaining musical sections and chooses the next musical section using Gaussian selection. The agent repeats this process using the previously selected musical section. The agent stops when a user-defined duration has reached.

ParamBOT ㉗ is a curator agent that generates musical sections of Moment form (Eigenfeldt, 2016a). *ParamBOT* initiates or stops other musical agents to create distinct musical sections. This implementation explores Stockhausen's idea of Moment Form in experimental music. A musical section in Moment form is free of the previous and next sections. At the beginning of each musical section, *ParamBOT* sets global parameters of *speed, activityLevel, voiceDensity, complexity, volume, consistency,* and *pitch*; and initiates other musical agents within the Musebot framework (Figure 9). We introduce the Musebot framework in Section 8.

5.1.2. Reactive musical agents with evolutionary computation

Evolutionary Computation (EC) is an abstraction of Darwinian evolution. EC implements *survival of the fittest* to find a solution to a given problem using a population of solutions. EC applies the genotype-phenotype dichotomy. A genotype is a representation of a solution (*phenotype*). A *fitness function* evaluates phenotypes (solutions) by assigning *fitness scores*. Genetic operators, *crossover, mutation* and *reproduction*, generate new solutions called *offsprings*. These offsprings go through a selection process to create next generations. An EC algorithm continues evolving a population until *stopping criteria* are reached (Sivanandam & Deepa, 2007). EC has been widely used in optimisation problems such as synthesiser preset generation (Tatar, Macret, & Pasquier, 2016). Although the problems in MuMe research do not necessarily involve optimisation, EC has also been widely used in the MuMe context (Miranda & Biles, 2007). In this section, we cover musical agents in real-world environments that use EC to generate musical material for the agent.

We begin by presenting three systems applying improvisation tasks. *GenJam* (28) is one of the early musical agent systems (Biles, 1994). *GenJam* implements a sub-branch of EC algorithms, called Genetic Algorithms (GAs) to generate melodies. *GenJam* improvises Jazz Music using an Interactive Genetic Algorithm (IGA) that generates musical phrases using a symbolic representation of music (MIDI). In IGAs, the user evaluates and scores all individuals in the GA population every generation. *GenJam* concentrates on jazz improvisation over a given chord sequence. Learning, breeding and demo are the three modes of *GenJam*. In the learning mode, a human listener gives real time fitness scores to *GenJam*'s musical phrases. The listener labels the *GenJam*'s current solo with the labels 'g' and 'b' that stand for 'good' and 'bad' respectively (Figure 10). In the breeding mode, *GenJam* implements genetic operators on the population of musical phrases. *GenJam* replaces half of the population every generation with new offspring using crossover and mutation. In the demo mode, *GenJam* improvises on a Jazz tune, using a chord progression file. Although Biles (1994) does not present *GenJam* as a musical agent, we can analyse the system architecture as a musical agent, with inputs of chord progression, rhythm section and human evaluation of the generated phrases, and with an output of solo jazz improvisation. Biles, who is also a trumpet player, presented *GenJam* in numerous concert venues as GenJam being another jazz player (Biles, 2013).

The following two systems that apply improvisation tasks imitate the style of human performers using EC:

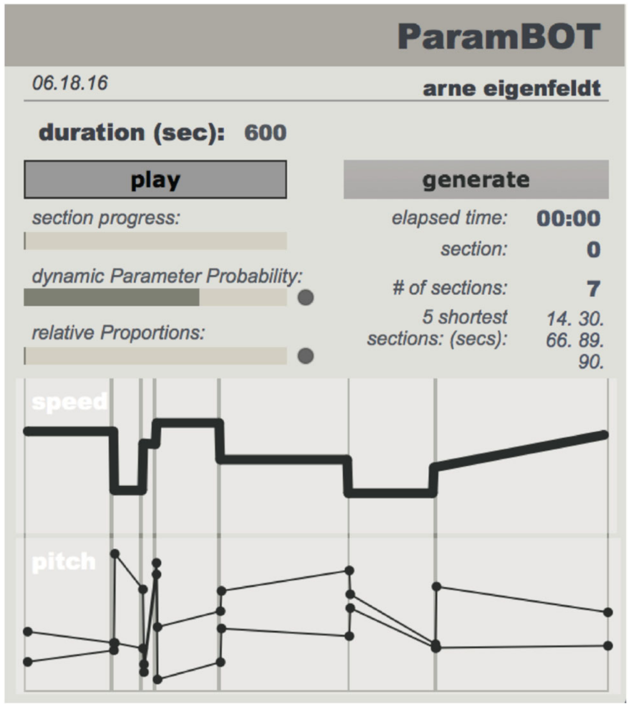
ParamBOT
06.18.16
arne eigenfeldt
duration (sec): 600
elapsed time: 00:00
section: 0
# of sections: 7
5 shortest sections: (secs): 14. 30. 66. 89. 90.

Figure 9. *ParamBOT*, a curator agent implemented using the Musebot framework in MAX (Eigenfeldt, 2016a).

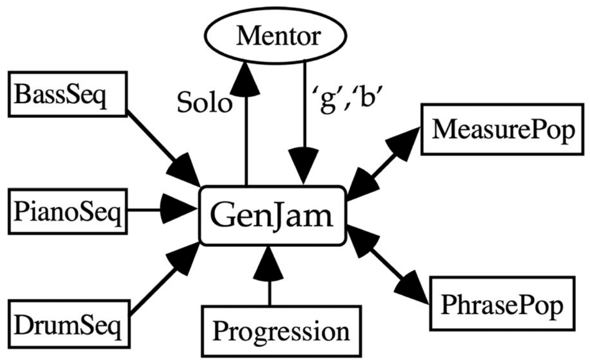

Figure 10. The block diagram of GenJam (Biles, 1994).

the system (29) and *Frank* (30). Yee-King (2007) study comes forward with a unique implementation of evolving timbres, rather than evolving symbolic representations of music. Yee-King (2007) presents a reactive musical agent (29) for live performances with human performers. This study also implements IGA with the Java programming language and embeds SuperCollider's *scsynth* framework for sound synthesis. The agent records the pitch and amplitude information of the live input to its memory to imitate the improvisation of other performers (Figure 11). The implementation includes two audio synthesis techniques: Additive Synthesis and Frequency Modulation (FM) synthesis. A control data of pitch and amplitude provided by the memory of the agent manipulates the parameters of these synthesisers. The IGA optimises the timbre of the audio synthesis to match the agent's timbre to the other performers. The author presents the initial results of this system, without recombination in the genetic algorithm. Hence, breeding is mutation only in the initial experiments. The author also comments on the improvisational skills of the agent, saying that the agent has a 'responsive feel' with 'unique dynamic sound '.

Frank (30) is a musical agent that evolves *lexemes* to imitate a human performer (Plans & Morelli, 2011). Lexemes are clusters of MPEG7 vectors[8] The clustering method of Frank is k-NN and the cluster locations are the centroid vectors. EC implementation includes two genders: *male* and *female*. *Frank* introduces the input frames as new female individuals to the population. During reproduction, the offspring gender is set to either male or female randomly. The authors clarify that female agents function as critic agents and the implementation of two genders introduces criticism to the system. The fitness function uses Euclidian distance between the input frame vector and an individual with an application of *imprecise pattern matching* with weight matrices. To implement imprecise pattern matching, the authors utilise a similarity threshold that female individuals use to choose a male individual to mate. The authors state that this balances coherence and novelty in the system. *Frank* plays back winner frames from the population.

The next two applications implement MAS with EC to generate rhythms. Gimenes, Miranda, and Johnson (2005) implement Dawkin's idea of memes in reactive musical agents. The system is called RGeme (31) and the MuMe application of interest is rhythm generation with symbolic representation. RGeme includes three types of agent tasks: *listening, practicing* and *composing*. The listening and practicing phases constitute the learning whereas composing is the generation. Each agent also has an evaluation algorithm to choose which music files are used in the learning tasks. During the learning, agents generate a *Style Matrix* in which the rhythmic memes are stored. The weights of rhythmic memes are determined by the number of times it is encountered and in which listening cycles it is encountered. To introduce temporality to the agent memory, rhythmic memes also lose weight if they are not encountered in later listening cycles. The authors also present an analysis of case studies in which agents are trained with Brazilian music composed by Chiquinha Gonzaga, Ernesto Nazareth, Jacob do Bandolim and Tom Jobim. The generation phase was not implemented in this version. In the following years, the generation phase was also implemented and this system is used as the brain for a robotics implementation (Gimenes, Miranda, & Johnson, 2007). However, we exclude robotics implementations in this review.

The latest version of *Kinectic Engine* (21) also applies rhythm generation with EC (Eigenfeldt, 2009, 2011). This study aims to generate rhythms that continuously evolve, rather than rhythms emerging, or appearing. Eigenfeldt (2009) states that EC provides a musical memory; and thus, introduces temporality. The system analyses a corpus of rhythms (MIDI files) offline. Moreover, *Kinetic Engine* analyses the individuals in the population on two musical dimensions: density (the number of events) and complexity (the degree of syncopation). EC in *Kinectic Engine* implements Roulette Wheel Selection, a well-known selection algorithm in EC (Sivanandam Deepa, 2007). The author mentions that *Kinectic Engine* includes a crossover-like breeding using a single parent; however, the details of this crossover-like breeding are not disclosed. The system also implements mutation. The population of rhythms is provided to all player agents. A player agent chooses individuals (rhythm patterns) in the population using user-set global density and complexity parameters (Figure 12). The player agent utilises a k-nearest algorithm to find rhythm patters with the user-given density and complexity values in the population. Eigenfeldt Pasquier (2009) present artistic implementations of the system along with the artistic evaluation of Kinectic Engine, concluding that following versions of the system should introduce 'intelligent' melody and harmony generation.

Aucouturier (2011) implements a multi-agent society (32) to evolve tuning systems. The fundamental frequency and the timbre are the global variables of the system. The agents include a dissonance calculation formula that is the parameterisation of the experimental Plomp-Levelt curves.[9] Each agent has a tuning system with the same number of notes. One agent listens (*tuner role*) while the other plays (*player role*). The tuner agent tunes its notes by minimising the dissonance between the player agent's note and its scale. There are two types of interactions between agents: *single note shift* and *drone shift*. In single note shift, the player agent chooses a random note to play and the tuner agent tunes one note that is chosen randomly. In drone shift, the player agent plays one note and the tuner agent tunes all notes in its memory by minimising the dissonance. The environment includes two types of timbres as global variables: harmonic timbre (where the partials are the integer multiplication of the fundamental frequency) and compressed timbre (where

[8] MPEG 7 is a standardisation of low-level feature calculation and thumbnailling for multimedia.

[9] Given a root note, Plomp-Levelt curves were proposed to calculate consonance or dissonance of any other note to the root note (Plomp Levelt, 1965).

the partials are spaced narrowly as stated by a geometric law).

5.2. Reactive musical agents in virtual environments

A recurrent theme in musical agent studies is the applications of self-organising agents that situate in virtual environments. The authors define the dimensions and properties of virtual environments. Systems generate music using the spatial orientation of agents and/or virtual encounters between agents. This data is mapped to parameters of audio synthesis, or symbolic representation of music. Hence, the complex behaviours in a virtual environment create music.

Bown, McCormack, and Kowaliw (2011) propose five elements of ecosystemic creative domains: *space, materials, features, actions* and *processes*. In ecosystemic creative domains, an agent situates in *space*, perceives the environment using *features*, performs *actions* using *materials* and the environment changes due to *processes*. In the following sections, we categorise musical agents in Virtual Ecosystems in two groups.

5.2.1. Multi-agent simulations with evolutionary computation

In this section, we cover musical MAS in which the system uses EC to evolve agents. Following, we present two implementations of melody generation. Todd and Werner (1999) present a system ㉝ that evolves monophonic melodies through agent interactions. The implementation is inspired by mating calls and singing rituals of wild animals in nature. There are two types of agents, referred by the authors as *male/composer* and *female/critic*. Male agents have a 32 note melody that spans two octaves. Each female agent includes a Markov model transition matrix indicating the probabilities of note transitions to rate the melodies of male agents. Each female listens to a subset of randomly selected male agents. After listening to all male agents in the subset, female agents choose one male agent to mate. The system uses crossover and mutation operators to generate offspring. The selection process of this system is not disclosed.

Similarly, *Living Melodies* ㉞ is a musical agent system that is also inspired by mating calls in nature (Dahlstedt & Nordahl, 2001). There are two genotypes in Living Melodies: sound and procedural genotype. Sound genotype dictates how an agent listens to other agents and how an agent generates sound. Procedural genotype designates how an agent interacts with and traverses in the ecosystem. The agents born when two agents mate, but there are no genders in the system. The system creates the genome of offspring using crossover and parents' genome. Moreover, each agent is born with an energy level. The agents loose energy points as they act within the environment. An agent dies when its energy level is below a threshold or a global, preset maximum life span has been exceeded. There are different configurations of sound mapping in the system. The agents generate mating calls as MIDI outputs using the information coded in their genome and communicating with other agents. The authors reported that the system can generate recurring patterns.

Martins and Miranda (2007) presented a system ㉟ that generates rhythmic phrases. This implementation focuses on the abstraction of music as a cultural phenomenon driven by social pressure (the system number 31 in Table 1). Although this study includes an A-life algorithm rather than EC, the ideas of survival of the fittest, breeding and assessment of a fitness score also appear in this implementation. The system includes a population of agents that are identical in the system architecture. The agents situated in a 2D space in which they interact with each other. Each agent has a memory of rhythmic phrases. During each interaction, one agent takes the role of *player* and the other takes the role of *listener*. As a consequence of the interaction between two agents, each rhythmic phrase of the player agent is given a popularity score by the listener agent. If the listener agent recognises a rhythm of the player agent, the listener agent gives a higher score to that rhythm, or vice versa. Moreover, the popularity of all rhythms drops by 0.05 after each interaction to introduce *aging* to the system. A transformation algorithm is applied to each rhythm that is shared between two agents during an interaction to foster novelty in the system. Martins and Miranda (2008) further analyse the system measuring the similarity of rhythms. The analysis includes size and complexity of an agent's rhythm memory, the similarity and clustering of agents, lifetime and novelty of generated rhythms. The authors state that the system exhibits 'the emergence of coherent repertoires across the agents in the society' in which the size of an agent's memory can be controlled using the popularity parameter.

Miranda, Kirke, and Zhang (2010) present a system that evolves expressive performance of music using an imitative multi-agent system, called *Imitative Multi-agent Performer* (*IMAP* ㊱). The authors define *expressive music performance* as the performance strategies that are not explained in the score, also known as the problem of *interpretation* in the context of MuMe. *IMAP* uses an *imitative model of behaviour transmission*, that is similar to the GA model of behaviour transmission. In both of these models, the algorithms generate a population of agents whose behaviours are defined by a genetic code. The difference between these two models is that the GA model uses a global fitness function whereas the imitative model has an non-global fitness function where each agent has a different fitness function. In the imitative model of behaviour transmission, an agent shares its behaviour to the other agents. Agents evaluate this behaviour using their fitness function, and if the behaviour scores high enough, the behaviour of the evaluating agent is updated accordingly. Hence, an agent has two functions: performance and evaluation. The parameters of the interpretation task are tempo and the loudness deviations. The fitness function is rule-based, implementing five rules of *performance curves, note punctuation, loudness emphasis, accentuation,* and *boundary notes*. The rules have weights that are particularly set for each agent. These weights make agent's fitness function unique.

Another implementation that combines biologically inspired algorithms with MAS is *RiverWave* (37) (McCormack & Bown, 2009). The researchers explore the idea of niche construction in ecosystem modelling to create digital art and music. Niche construction is the phenomena of organisms establishing a more habitable environment for their offspring, which can also be approached as a process that precedes the evolution. The cooperation between agents becomes more prominent since parents aim for more habitable environments for their offspring. The musical agent system, *RiverWave* is a one dimensional, toroidal ecosystem that controls an additive synthesiser. Each agent location determines the frequency of the oscillator. Each agent has a height variable and agents affect the height of the neighbouring agents. The height parameter of an agent is mapped to the amplitude of the oscillator.

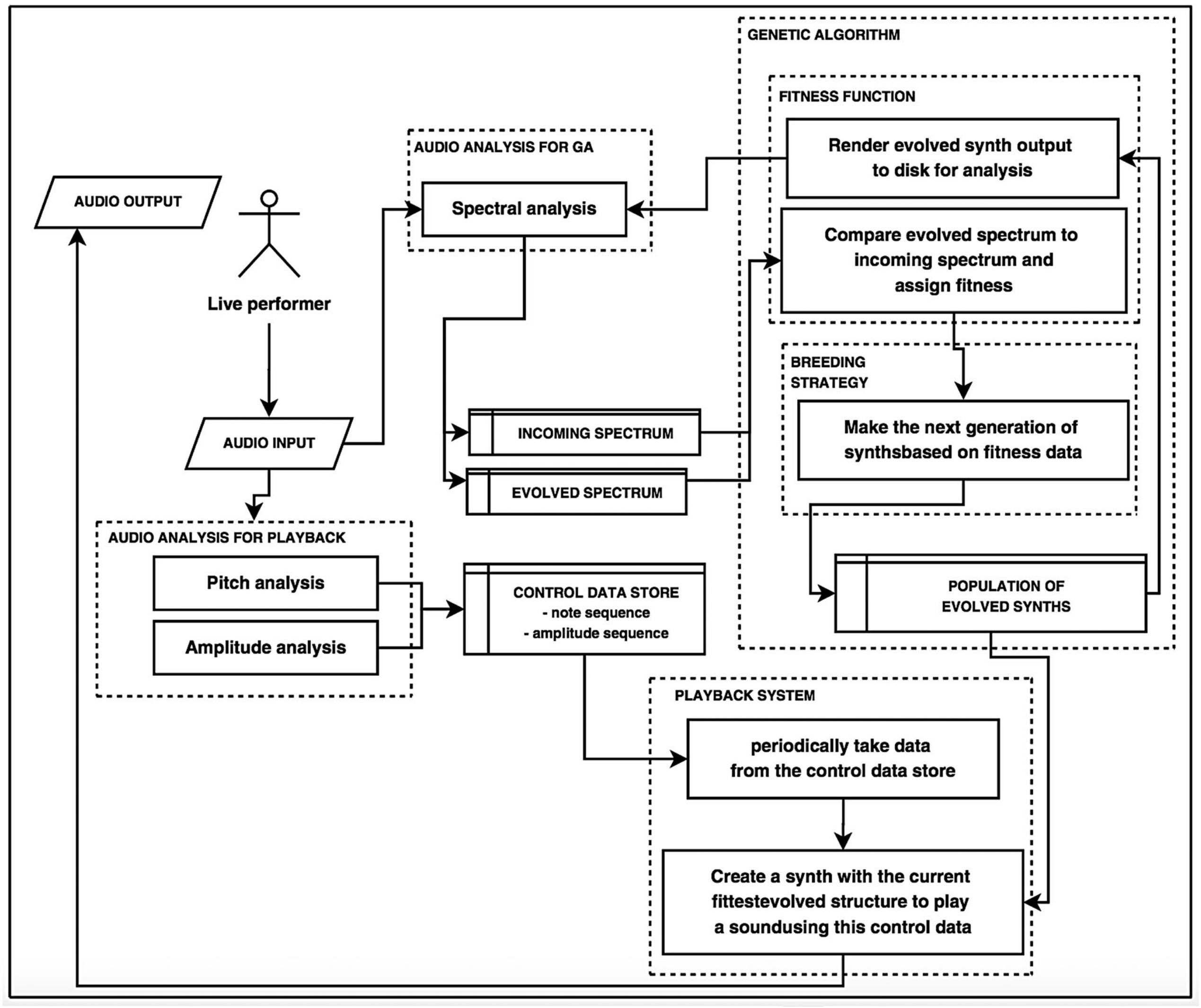
GENETIC ALGORITHM
FITNESS FUNCTION
Render evolved synth output to disk for analysis
Compare evolved spectrum to incoming spectrum and assign fitness
BREEDING STRATEGY
Make the next generation of synthsbased on fitness data
POPULATION OF EVOLVED SYNTHS
AUDIO ANALYSIS FOR GA
Spectral analysis
AUDIO OUTPUT
Live performer
AUDIO INPUT
INCOMING SPECTRUM
EVOLVED SPECTRUM
AUDIO ANALYSIS FOR PLAYBACK
Pitch analysis
Amplitude analysis
CONTROL DATA STORE
- note sequence
- amplitude sequence
PLAYBACK SYSTEM
periodically take data from the control data store
Create a synth with the current fittestevolved structure to play a soundusing this control data

Figure 11. The system architecture of Yee-King's (2007) musical agent.

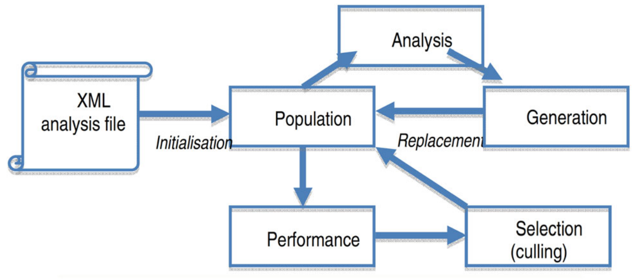

Figure 12. The block diagram of Kinectic Engine version 3 (Eigenfeldt, 2009).

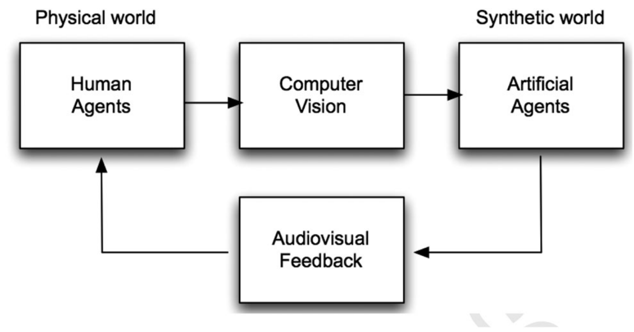

Figure 13. The system design of *Petri* (Beyls, 2012).

Petri (38) is an interactive audio-visual system that utilises a virtual environment and the real-world interactions (Beyls, 2012). The reactive agents situated in a virtual environment while the parameters of virtual environment change with a computer vision input (Figure 13). A webcam input is processed with a computer vision algorithm that provides five visual features to the virtual environment. These features define attraction points in the virtual environment. The agents move closer to these attraction points. There is also a life cycle that each agent goes through. New agents are created closer to the attraction points. The reactive agents have genders and communicate with each other. The communication between neighbouring agents results in the sound generation. When agents decide to generate sound, the location of the agent defines the synthesis parameters. The 2D virtual environment is mapped to the FM synthesis parameters of carrier frequency and modulation frequency.

5.2.2. *Multi-agent simulations with ecosystemic approaches*

Blackwell Young (2004) present two applications of swarming in symbolic music generation. Swarming is a multi-agent behaviour that is inspired by the behaviours of animal herds like birds, fishes and insects. The behaviours emerge as a result of four rules: agents try to move closer to neighbouring agents, neighbouring agents avoid collisions, all agents try to match velocity including the direction, all agents try to move towards attraction points. Likewise, self-organisation has four components: positive feedback, negative feedback, amplification of fluctuations and multiple interactions. *Swarm Music* (39) and *Swarm Granulator* (40) are ecosystemic reactive agents where the agents situated in a virtual environment. In *Swarm Music*, the authors propose a mapping between the spatial locations of agents and symbolic music parameters of pitch, loudness, inter-onset interval, duration, chord number and sequence number. The authors also propose the idea of using two swarms for symbolic music generation where the spatial locations of one swarm are the attraction points of another. In *Swarm Granulator*, the agent records the human performer in an audio buffer while calculating the audio features of the pitch, amplitude, duration and duration between successive sound-events. The swarming outputs six parameters audio buffer transposition, amplitude, duration, the time between successive grains, grain attack and decay time.

Ando and Iba (2005) propose a musical agent system (41) including a virtual environment in the application of extended instrument design. Although the authors claim that the system is a cellular automata implementation that includes a MAS, the details of the system design is not disclosed. Ando and Iba (2005) say that the states of agents in the cellular automata change according to some pre-defined rules while a human performer plays a MIDI keyboard. The details of these rules are also not presented in the study.

McCormack, McIlwain, Lane, and Dorin (2007) propose a unique idea of using a 2D virtual environment with musical agents as a dynamic graphic score that generates music. The name of this framework is *Nodal* (42) and available as a commercial software.[10] Users create a virtual environment using *nodes, edges, node traversals,* and *player agents*. Users put nodes in the environment and create connections between these nodes. These connections are called *edges*. Player agents traverse the nodes and edges. Each time a player agent reaches a node, the agent plays a MIDI note and changes its state variables (lists of pitch change, note-on, and note duration, MIDI instrument). The authors also give examples of bi-directional and asymmetric cycles, and cyclic pitch phasing as the examples of emergent behaviours in *Nodal*.

OSCAR (43) is a MAS with reactive agents situating in a virtual environment (Beyls, 2007, 2011). This application focuses on the problem of generating non-idiomatic improvisation (a.k.a. free improvisation) with the symbolic representation of music. This study focuses on *autopoiesis*; that is ‘the continuous creation of new

[10] http://www.nodalmusic.com/

answers while facing an unpredictable environment'. Agents situated in a 2D environment having parameters of *physical position, energy level, distance of communication, distance of neighbourhood, activation, orientation, affinities* and *personality dataset* of *pitch intervals, durations* and *velocities*. The system tries to minimise overall social stress by using affinities between agents. The system generates musical output using the histogram of agent communications on each iteration. An agent that initiates a musical event generation chooses one of two methods: *contraction*, and *expansion*. Contraction generates a single musical event using a set of events, whereas expansion generates supplementary events using a single source event. The authors also presented three experiments with the systems to show emerging patterns. The system presented periodic patterns running over longer durations as well as complex behaviours.

Eigenfeldt and Pasquier (2011b) use Concatenative Synthesis with an ecosystemic MAS. The authors proposed the idea of generating music through consumption of virtual food in a virtual environment. The system is referred as *Shoals* (44) that is a part of series of generative music systems, called *Coming Together*. The system uses Concatenative Granular Synthesis with CataRT, an external library that is available in the visual programming language MAX. The real-time audio feature extraction of audio input creates food in the virtual environment that agents situate. The agents can move within the virtual environment. As an agent finds and consumes food, the consumption is sonified using CataRT. The agents are randomly initialised with synthesis parameters of *grain duration, delay between grains, amplitude, offset into the sample, phrase length, pause between phrases, phrase type, output*, and with MAS parameters of *acquiescence* (desire to stay at the location of a food source), and *sociability*. Agents have a histogram of encountered food sources. This histogram affects the decision of an agent's movement. The audio input is recorded into an audio buffer when it is not silent. The existence of sound in the audio input also creates excitement in the virtual environment and the agents start moving at faster rates. There is also communication between the agents. When an agent finds a food source, the agent shares the location of the food source. The agents die if they cannot locate a food source for a certain time. The death agents are reincarnated after a variable duration that is between 5 and 60 s. The agents create social networks by sending 'friend requests' and using the sociability ratings. The agents can also leave a network to join a bigger network. Eigenfeldt and Pasquier (2011b) stated that even when the agents find a food source, and the network becomes static, the social networking still create dynamic behaviours.

Beyls, Bernardes, and Caetano (2015) presented a MAS implementation focusing on a cultural phenomena. The system, *earGram Actors* (45) is based on the *Actor* model (Beyls, 2011) which is a derivation of the *Party Planner* model (Gold & Maeda, 2007). In the actor model, the reactive agents aim to be as close as possible to the agents that they like (and vice versa) to minimise the social stress of the society. The actor model works on two dimension *affinity* and *sensitivity*. Affinity is pre-set and dictates the attraction of an agent towards another. Sensitivity sets a distance threshold to apply affinities of agents. This simple abstraction of a virtual society creates complex movements of agents in the virtual environment. This MAS implementation uses a hybrid audio corpus. The corpus consists of 200 ms long audio samples, and the samples are mapped to a 2D space using dimensionality reduction on a set of audio features, including noisiness, pitch, brightness, spectral width and sensory dissonance. Then, this 2D audio feature space is mapped to the 2D virtual environment. Hence, the movements of agents in the environment create musical output with the concatenative synthesis.

Likewise, *pMIMACS* (46) is an ecosystemic MAS that generates symbolic music with interpretation (Kirke & Miranda, 2011). Each agent has the same architecture and has a tune in the memory. The agent with similar tunes performs to each other during each cycle. When an agent performs to another, the listening agent learns the interpretation of the other agents tune. There are four dimensions of the interpretation in *pMIMACS*: accuracy/tempo, excitation state, key and microerrors.

Al-Rifaie and Al-Rifaie (2015) present a MAS (47) generating musical melodies with symbolic representation. This implementation also uses a swarm intelligence algorithm, *Stochastic Diffusion Search (SDS)* that is inspired by one species of ants, *Leptothorax acervorum*. In SDS, the agents situated in a search space and communicate with one another directly. SDS has two phases: *test* and *diffusion*. During the test phase, each agent implements exploitation. If an agent finds a better solution, the agent is considered *happy*. During the diffusion phase, each agent talks to another agent that is randomly chosen. If an unhappy agent talks to a happy agent, the unhappy agent is considered *lucky*, and the happy agent shares its location with the lucky agent. In each iteration, the number of local unhappy, and lucky agents are stored for the sonification. The focus in this implementation is generating musical scores that sonifies the agent communication. The system uses plain texts as an input, and the letters are mapped to pitch, note duration and dynamic. The population size is 20, and the number of iterations (episodes) is 10. The authors exemplified this implementation with a melody generated by the text ‘hello music sds welcome to the reality’.

Similarly, Gimenes and Miranda (2011) applied cultural ecosystem approach in the design of *Interactive Musical environments* (*iME* (48)). *iME* concentrates on monophonic melody generation. The system design applies the ideas of ‘memetics’. Memetics (as in genetics) is the idea that the development of cultural organisms is through the smallest functional units that are *memes* (as in genes). The authors propose the term *ontomemetic* (inspired by ontogenetic) that is ‘the sequence of events involved in the development of individuals musicality’. The authors also point out the characteristics of ontomemetic systems:

(1) Modelling cognitive and perceptive abilities of humans,
(2) Using the interaction between artificial entities to create emergency
(3) Modelling interactivity through the communication between artificial entities
(4) The availability of comparison of different musical styles generated by an application of ontomemetics

iME applies ontomemetics to monophonic melody generation using a feature extraction on MIDI data. The features are melodic direction, melodic leap, inter-onset interval, duration, intensity and vertical number of notes. *iME* has a virtual ecosystem in which agents listen each other. Each agent has the same architecture. One agent takes the role of *player* whereas the other takes the role of *listener*. The agents have two types of memory: long-term and short-term. The long term memory stores all unique memes that the agent encounters. Each meme has a connection pointer that is the index of the successor meme. Each meme has a weight that increases as the agent encounters a meme more. In that sense, this structure resembles a first-order Markov model. Short-term memory only saves a user-defined number of memes that the agent encountered the latest. The system is capable to generate music as using solo agents, as well as collective improvisation. The generative algorithm includes a pre-set *compositional and performance map* that guides agents to choose memes.

Our survey of cognitive and reactive musical agents finishes here. We continue by reviewing musical agents that combine cognitive and reactive modules together in their system design.

6. Hybrid musical agents

Hybrid agent architectures include both reactive and cognitive modules together. Following, we discuss four subcategories of hybrid musical agents.

6.1. Hybrid musical agents using statistical sequence modelling

A recurrent theme in hybrid musical agent studies is the implementation of statistical sequence modelling algorithms such as Incremental Parsing (IP), Probabilistic Suffix Trees (PSTs), Factor Oracles (FOs), Partially Observable Markov Decision Processes (POMDP), Variable Markov Models (VMM), Hidden Markov Models (HMM). Many systems presented in this section use Markov Decision Processes or Markov Models or Markov Chains.

Markov Models are finite state machines that encodes patterns os transitions between discrete states using the Markovian assumption. The Markovian assumption of an N^{th} order Markov model is,

$$P(s_t|s_{t-1}, s_{t-2}, \dots, s_1) = P(s_t|s_{t-1}, \dots, s_{max(t-N,1)}). \tag{1}$$

The order of a Markov model dictates how many previous states to be considered to predict and generate the next state. Moreover, the conditional probabilities of the transitions depend on the observed number of transitions between the states.

The environment is discrete and stochastic in Markov Models (Puterman, 1994). The environment is stochastic because given a particular state, the resulted next state of an agent is not certain. Specifically in Markov Decision Processes, we can define a probability function $p(s'|s, a)$ where s is the current state of an agent, and a is an action. *Reward* function, $r(s, a)$, evaluates an action, a, performed in a particular state, s. *Policy* (or *decision rule*), d, is an assignment function, $d : S \rightarrow A$, where S is the set of all possible states and A is the set of all actions that are available to an agent. Hence, a *policy* specifies which actions should be performed in which states. In Markov Models, *value iteration* algorithm defines how to find an optimal policy (Puterman, 1994). Moreover, Markov Models can apply learning. Agents generate the transition probabilities between states during learning. For example, Martin, Jin, van Schaik, and Martens (2010) implement Partially Observable Markovian Decision Processes (POMDP) (49) in the design of a hybrid musical agent that listens to human performers (the system (74)). The system generates melodies in the key of the human performer using tonal harmony theory in music.

The initial experiments on using statistical sequence modelling algorithms for music focused on how to create optimal tree form representations using compression algorithms for online applications of music. Two early works (Assayag, Dubnov, & Delerue, 1999; Dubnov, Assayag, & El-Yaniv, 1998) concentrated on generating sequences of melodies using Lempel-Ziv compression algorithm. Later, their work evolved into two interactive musical agents, the Continuator and *OMax*. The Continuator (50) is a well-known musical agent on the problem of musical style *imitation* (Pachet, 2003, 2004). The *Continuator* study proposes the MuMe problem of *continuation*, that is, continuing a performance in the style of the performer when the performer stops. Using symbolic representation of music (MIDI), Pachet (2004) introduces hierarchy and bias to Variable-Order Markov Models to handle the polyphony, noise and arbitrary rhythmic structures in the input. The agent architecture consists of two parts: *analysis* and *generator*. The analysis module has three submodules: *phrase end detector* (adaptive temporal threshold mechanism), *pattern analyser* (the generation of Variable-Order Markov Model) and *global property analyser* (number of notes per second, tempo, meter and overall dynamics). The Continuator has two modes of interaction: *question and answer* and *collaboration*. What Pachet (2004) refers as question and answer are *call and response*, a well-known improvisation setting in the context of jazz. In the collaboration mode, the *Continuator* implements accompaniment by listening to a human performer in real-time and adapting the generated output in parallel with the human performer's style. Moreover, Pachet (2003) proposes three implementations of *Continuator*. First, a musician can play with a Continuator trained on a famous musician's performance. Second, multiple musicians can have multiple Continuators trained on different musical performances. Musicians can also have Continuators trained on the same corpus. Third, the *Continuator* can extend a soloist's capability or accompany a soloist by training on a corpus of chord sequences.

Beatback (51) (Hawryshkewich, Pasquier, & Eigenfeldt, 2010) also focuses on the tasks of accompaniment, and call and response by implementing Variable Order Markov Models to generate musical rhythms. The system represents rhythm sequences in three dimensions: inter-onset time difference, velocity (MIDI) and drum type (instrument). *BeatBack* focuses on two musical applications: accompaniment, and call and response. Hawryshkewich et al. (2010) present a technique called *Drum-kit Zoning* to use *BeatBack* as an *Expanded Instrument System* that expands the performance of a drummer. The pattern generation in *BeatBack* has two modes: *query* and *build*. In the Query mode, *Beatback* uses the last rhythmic pattern of its input to search for and assign probabilities to possible next patterns. In the Build mode, *Beatback* generates a rhythmic pattern using the probabilities generated by the Query mode.

Ringomatic (52) is another musical agent that generates rhythm accompaniment (Aucouturier Pachet, 2005). The authors implement two classification tasks to

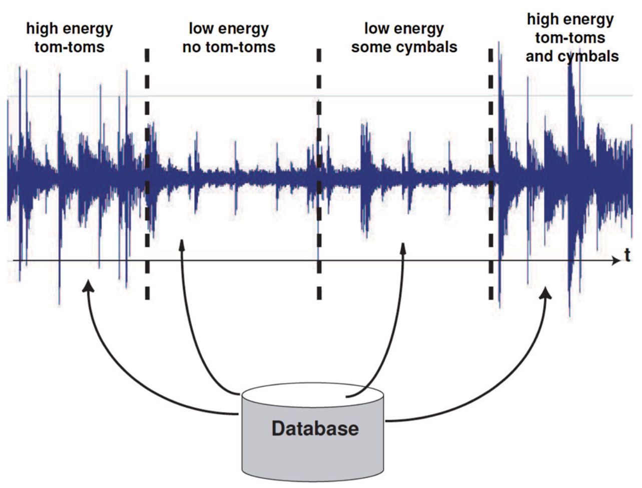

Figure 14. Energy-based generation in Ringomatic (Aucouturier & Pachet, 2005).

automatically generate a hybrid corpus. The first task is to find solo drum sections in recordings and the second task is to label them with three energy levels of low, medium and high (Figure 14). *Ringomatic*'s architecture includes constraint-based concatenative synthesis to generate audio. Ringomatic sets the constraints of energy, onset density and pitch. The authors propose a new technique called incremental adaptive search that is an implementation of local search techniques in constraint satisfaction problem. The constraints are introduced to the system as cost functions. There are two types of constraints: local and global. Local constraints look only at the current state to predict a next state whereas global constraints include past states in the cost function. Aucouturier and Pachet (2005) also present an analysis of the system in a duo with a human performer playing MIDI keyboard.

Factor Oracle (FO) is a finite state automata that is a variation the suffix tree. FO represents substrings and patterns in a sequence, that is, at least all *factors* of a sequence. FO has three types of links: internal links, external links and suffix links. Internal links are forward links between successive states. External links are forward links that jump longer than successive states. Suffix links are backward links that point the longest repeating factor in the previous states. FO allows incremental learning, and learning is linear in time and space (Lefebvre & Lecroq, 2002). Assayag and Dubnov (2004) compare IP, PSTs and FOs for the symbolic sequences of music. Assayag and Dubnov (2004) conclude that FOs suit the best to satisfy incremental and fast online learning, time-bounded generation of musical sequences and implementation of multi-attribute models to deal with the multi-dimensionality of music. Within the last two decades, many studies implemented

FOs in musical agents (Assayag, Bloch, Chemillier, Cont, & Dubnov, 2006; Assayag & Dubnov, 2004; Donze et al., 2014; Dubnov, Assayag, & Cont, 2007; Einbond, Borghesi, Schwarz, & Schnell, 2016; Franois, Chew, & Thurmond, 2011; Franois, Schankler, & Chew, 2013; Lévy et al., 2012; Lynch, 2014; Nika & Chemillier, 2012; Nika, Chemillier, & Assayag, 2017; Nort, Oliveros, & Braasch, 2013; Surges & Dubnov, 2013; Valle et al., 2017; Wang & Dubnov, 2014).

Assayag et al. (2006) present a framework to implement musical agents with FOs, called *OMAX* (54). OMAX uses a FO based real-time machine improviser scheme (Assayag & Dubnov, 2004). Assayag et al. (2006) also propose two unique implementations of FOs: one with reinforcement learning, and the other with meta-level learning. First mentioned by Dubnov and Assayag (2005), *OMAX* listens to a performer, and learns the style of the performer using FO (Figure 15). The problem of *style imitation* is well-known in MuMe field (Pasquier et al., 2017). The agent generates by using navigation strategies combined with the links and factors within the FO model, browsing the model using diverse navigation stratgies, and renders these sequences sonically. Thus, the generated material is recombination of musical material in the agent's memory. The agent utilises polyphonic pitch duration slices with MIDI for the musical applications with symbolic representations, and real-time recorded audio segments in the case of audio (Lévy et al., 2012). Lévy et al. (2012) implement *OMAX* in MAX 5, including pitch estimation, and spectral clustering with Mel Frequency Cepstral Coefficients (MFCCs) and Fast Fourier Transform (FFT) in the analysis module. Ongoing artistic use of *OMAX* appears in two context, duo with a human performer playing an acoustic instrument, and control of an *OMAX* musical agent by an electronic musician (Lévy et al., 2012). The I/O of *OMAX* can be symbolic representation of music, or audio signals, or video signals (Bloch, Dubnov, & Assayag, 2008; Lévy et al., 2012).

The hybrid musical agent architecture (55) of Cont, Dubnov, and Assayag (2007) also implement FOs with an anticipatory model of musical style imitation with collaborative and competitive reinforcement learning. The authors use *multiple viewpoints* (Conklin, 2013), and there are four factor oracles trained for musical dimensions of *pitch, pitch contour, duration,* and *duration ratio.* The system can be used in two modes: *interaction* and *self-listening*. In the interaction mode, the agent listens to another agent (or human performer) whereas the agent listens to its own audio output in the self-listening mode.

Collins (2008) also presents a musical agent, called *Improvagent* (56) that uses reinforcement learning with symbolic representations of music. Using the MIDI input, the agent computes a set of onset, pitch, and rhythm

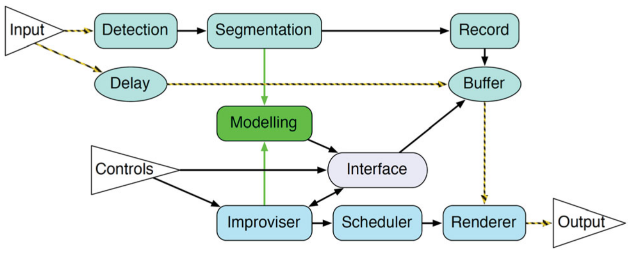

Figure 15. The block diagram of *OMAX* (Lévy, Bloch, & Assayag, 2012).

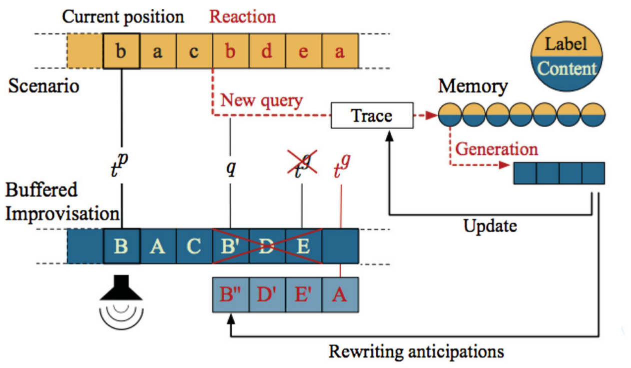

Figure 16. The improvisation renderer in Improtek (Nika, Bouche, Bresson, Chemillier, & Assayag, 2015).

features as well as higher level features such as key, pitch class, expressiveness and density. *Improvagent* treats input frames as the states of the environment. The system clusters environment states using k-nearest neighbours with Euclidian distance. The agent also updates its database in real-time. The included reinforcement algorithm is Sarva[11] Improvagent generates the audio using the concatenative synthesis.

Improtek (Nika et al., 2015; Nika & Chemillier, 2012; Nika et al., 2017; Nika, Echeveste, Chemillier, & Giavitto, 2014) builds upon OMAX, and implements *OMAX* with an introduction of tempo, beats, harmonisation, and arrangement. *Improtek* system uses three Factor Oracles for improvisation, harmonisation and arrangement, using the symbolic representation of music (MIDI) for the applications of improvisation and accompaniment. Nika et al. (2017) further develop *Improtek* by introducing a *scenario/memory generation* model. Figure 16 shows the guided improvisation in *Improtek* with a scenario and memory. The authors use any alphabet in the musical context, such as audio, MIDI, or sound synthesis parameters. A symbolic sequence of labels defined with the alphabet is the *scenario* whereas a sequence of musical contents labelled using the alphabet is the *memory*. The improvisation is guided by the

[11] The details of Sarva is available in the book on reinforcement learning by Sutton and Barto (1998).

scenario using two strategies: *anticipation* and *digression*. Using anticipation, *Improtek* searches the memory for a starting sequence. The constraint is that the starting sequence matches the future labels that follow the current state of the scenario. Digression strategy ensures that *Improtek* finds a continuation sequence in the memory. This continuation sequence matches both past and future states of the current state of the scenario. The implementation consists of three agents: *improvisation handler*, *dynamic score*, and *improvisation renderer*. The improvisation handler is a reactive agent that implements the guided music generation using a scenario and a memory. The dynamic score handles perception of *Improtek*'s environment. The improvisation renderer conducts the output generation using the content generated by the improvisation handler. Nika et al. (2017) points out two cases of performance with Improtek: human and machine, and machine only. When Improtek is trained on audio, the agent conducts online audio generation using a phase-vocoder. Hence, Improtek can sample live audio and apply time stretching, pitch-shifting and crossfade transformations in real-time to temporally and harmonically align the generated improvisation with a pre-defined scenario.

In parallel with the studies on *OMAX* framework, Dubnov et al. (2007); Dubnov, Assayag, and Cont (2011) present *Audio Oracle* [58]. Inspired by FOs, *Audio Oracle* is an algorithm that detects repeating sub-clips of variable length in audio data. Dubnov et al. (2011) define these sub-clips as *audio factors*. Similar to FO, *Audio Oracle* analyses an audio file as a string of audio feature vectors. The user can choose different audio features (or combinations of audio features) to train an Audio Oracle. Forward links in *Audio Oracle* refers to the states that generate similar patterns by continuing forward whereas backward links correspond to the states sharing the largest similar sub-clip in an audio file. *Audio Oracle* uses Euclidian distance between audio features to decide if two states belong to the same class. The user sets a similarity threshold, and if the Euclidian distance between two states is below the threshold, those states are accepted as equivalent. High similarity threshold means that distant states are more likely to be labelled with the same class, thus decreasing the size of the alphabet. Furthermore, Dubnov et al. (2011) introduce automatic threshold selection for the Audio Oracle using the notion of *Information Rate (IR)* in Signal Processing (Dubnov, McAdams, & Reynolds, 2006). AO uses the threshold that gives the highest information rate.

Surges and Dubnov (2013) further developed Audio Oracle studies by introducing a system for music analysis and machine improvisation, called *PyOracle* [59] (Figure 17). Similar to *Audio Oracle*, *PyOracle* includes an off-line learning that inherits signal complexity and familiarity analysis. Surges and Dubnov (2013) relate complexity and familiarity to aesthetic appreciation with Birkhoffs idea of aesthetic measure (Rigau, Feixas, & Sbert, 2008). Birkhoff defines aesthetic measure as the ratio between the order and the complexity. Audio Oracle uses IR to balance between the order and the complexity. IR measures the reduction of a signal's uncertainty using signal's past values. Surges and Dubnov (2013) stated that low IR refers to higher complexity and lower order whereas high IR corresponds to lower complexity and higher order. *Audio Oracle* uses IR measure to set the uniqueness distance threshold between the states. Surges and Dubnov (2013) aim for the highest IR in the implementations to extract the musical form information of a signal during the PO's learning process.

Building on the previous studies of FOs, *Audio Oracle*, and PO; Wang and Dubnov (2014); and Arias, Desainte-Catherine, and Dubnov (2016) introduce another musical agent system using *Variable Markov Oracles* (VMO [60]). VMO allows adaptive symbolisation of audio features to provide representation of higher musical structures. The system implements *Petri Net* graphical language for concurrent and distributed system design, *PyOracle* to create *Audio Oracles*, and I-score (Baltazar, de la Hogue, & Desainte-Catherine, 2014) to control generated models with graphic scores. The authors mention that the previous studies on musical implementation of Factor Oracles have been criticised by not representing higher musical structures, and this study addresses the representation of higher musical structures using *Petri Net*. Arias et al. (2016) propose that a possible next step for the development of this system is the introduction of *scenario/memory generation* model, presented by Nika et al. (2015).

Freely Improvising, Learning and Transforming Evolutionary Recombination (*FILTER*) (Nort et al., 2013) is a musical agent that combines FO and a type of Markov Models, called Hidden Markov Models (HMMs). HMMs are widely used to model temporal discrete sequences. HMMs consist of hidden states and observed states. The number of hidden states can be different than the number of observed states. The transition matrix is the likelihood of transitions between hidden states. The observation (or emission, or confusion) matrix is the likelihood of observations given a hidden state. HMMs have three applications: evaluation (likelihood of an observed sequence given an HMM), decoding (the sequence of hidden states that most likely to generate the observed sequence) and learning (generating an HMM given a sequence of observed states).

The musical application of *FILTER* is free improvisation. *FILTER* implements style imitation based on unsupervised learning. The learning applies Smalley's approach on textures and gestures in Electro-acoustic Music (Smalley, 1997). Using an inter-onset threshold, *FILTER* samples the audio input of the last *N* seconds and the memory encodes the temporal changes of audio features. If the recorded sample is dissimilar to the anything in the current memory, it is added to system's memory. *FILTER* includes sonic gesture and texture analysis. *FILTER* applies continuous gesture recognition method proposed by Bevilacqua et al. (2009) to learn sonic gestures of the input using Linear Predictive Coding (LPC), Mel-Frequency Cepstral Coefficients (MFCCs), autocorrelation coefficients and YIN algorithm features (frequency, energy and periodicity). The gesture recognition algorithm combines HMM with dynamic time warping. The system can learn a dictionary of gestures either offline using a corpus or online by listening to the input. The gesture recognition algorithm outputs the likelihood of gestures. Using the likelihood, *FILTER* can perceives the level of deviation from the current gesture of the input. The system also inherits a non-linear time-frequency analysis called intrinsic mode function to comprehend the sonic texture of the input. *FILTER* includes two types of memory: *semantic* and *episodic*. The semantic memory is the dictionary of distinct gestures whereas the episodic memory applies FO to learn temporal structures of the input. *FILTER* also applies a mutation only Genetic Algorithm (GA) for the adaptive goal decision process. The system introduces adaptivity by mapping the gesture/texture likelihood values to the fitness of GA.

Lastly, *SpeakeSystem* (62) is a musical agent with Variable Markov models (VMMs) (Yee-King d'Inverno, 2016). The agent uses FM synthesiser to generate audio. The modulation index of the synthesiser changes as the length of sequences generated by VMMs varies. The authors stated that using two VMMs, where one VMM handles the rhythm and the other focuses on the pitch, generates more varied output, comparing to the case with one VMM.

6.2. Hybrid musical agents combining statistical sequence modelling with rule-based models

Martin, Jin, and Bown (2011) present a framework for non-technical users to design musical agents. This framework, called *The Agent Design Toolkit* (ADTK (63)) implements the ideas of interactive machine learning in musical agents. ADTK consists of three elements: a set of recorded performance variables, a set of probabilistic temporal models and a set of rules defining the relation between performance variables. The framework uses VMMs for probabilistic temporal models and association rule learning (ARL) algorithms for automatic rule generation using the recorded performances. Following this study, Martin, Jin, Carey, and Bown (2012) introduce ADTK to Ableton Live, a well known Digital Audio Workstation (DAW). Martin, Jin, Carey, et al. (2012) conducted two case studies on ADTK, designing a musical agent that improvises electro-acoustic music, and a musical agent generating *Drum and Bass* music. Martin, Jin, and Bown (2012) mention a possible computational complexity problem with the initial versions of ADTK. The automatic rule generation solves the constrained satisfaction problem using ARL algorithms. However, it is not possible to know how long ARL algorithms take, and how many solutions the algorithm produces. This makes the systems designed with ADTK framework susceptible to bugs in real-time performances. To address this computational complexity problem, Martin, et al. (2012) propose binary decision diagrams (BDDs). Although the introduction of BDDs does not completely solve the computational complexity problem, the designer can examine if an agent is capable of real-time performance before the performance. Thereby, the system is no longer susceptible to bugs in real-time performances. Martin, Jin and Bown (2012) also compare BDDs-based ADTK to the initial version of ADTK with ARL, concluding that the parameter update duration was more predictable in BDDs-based implementation than the ARL-based implementation. Martin and Bown (2013) also demonstrate ADTK on style imitation. Bown and Martin (2013) mentioned that the musicians could control the agents designed using ADTK. Hence, these agents stand somewhere between an extended instrument and an autonomous performer.

CinBalada (64) is another multi-agent system that combines statistical sequence modelling with rule-based models (Sampaio, Ramalho, & Tedesco, 2008). The system generates polyphonic rhythmic sequences as the symbolic representation of music. Sampaio et al. (2008) are inspired by the music styles with an emphasis on the rhythm such as taiko, pungmul, samba batucadas and maracatu. *CinBalada* includes three rhythm representations of Time Unit Box System, Polygonal Representation, Time Elements Displayed as Squares to calculate rhythmic measures of offbeat-ness, evenness and rhythmic similarity as chrotonic distance. *CinBalada* includes these rhythmic measures in the evaluation functions. *Cinbalada* is a homogeneous MAS with multiple roles. There are multiple *rhythmic roles* that each agent can choose. The number and the type of rhythmic roles depend on the implemented musical style. For example, a Batacuda implementation has three roles of *base, complementary base and solo*. The evaluation functions also change depending on the implemented style. Within a bar, agents in *CinBalada* negotiate what to play in the following bar. The agents share their rhythmic patterns with the other agents. *CinBalada* outputs only the patterns that score the highest on the evaluation functions.

6.3. Hybrid musical agents with artificial neural networks

Artificial Neural Networks (ANN) is a set of Machine Learning algorithms. ANN algorithms are inspired by the theories of neuron activation and sensory data processing of neural systems in nature. ANN has been applied to the Machine Learning problems of classification and linear regression (Mitchell, 1997).

The *Reactive Accompanist* (65) is the first musical agent system that implements Subsumption architecture, including three ANNs in different layers (Bryson, 1995). Subsumption architecture (mentioned in Section 5) implements a hierarchical set of rules in which lower layers have higher priority, or vice versa. Reactive Accompanist is a mono-agent system with audio input and symbolic output (MIDI). There are three layers in the architecture: *pitch, chord* and *time*; ordered from the lowest to the highest layer respectively (Figure 18). The pitch layer has two modules. The first one implements Fourier transform and outputs frequency-gain pairs. The second module is an ANN with supervised learning. The input is frequency-gain pairs whereas the output is pitch classes. The chord layer is also an ANN. The input is pitch classes and the output is predefined chords. The highest layer in the hierarchy, time has four modules of *thresh, beat, timed*, and *change*. The first three modules handle rhythm. Beat module implements tempo estimation with ANN and change module handles chord changes. The application of this system is accompaniment of live input. Bryson implemented the system in the first half of the 90s when Fourier transform calculation was still too computationally complex for online applications. Because of the Fourier transform calculation in the pitch estimation, this system works offline.

Another musical agent with ANNs is *NN music* (66) (Young, 2007). The architecture is an implementation of the musical agent framework, *PQf* proposed by Blackwell et al. (2012), where *P* implements listening and analysis, *Q* handles performing/synthesis and *f* conducts patterning, reasoning, or generative functions. NN music includes two analysis functions in P module: *parameterisation of pitch characteristics* and *statistical representation of musical behaviour*. These two analysis output two independent state representations: a set of recently identified pitches and a set of statistical representation of audio features computed over 50 ms audio frames. The statistics are calculated with a varying window of 5–30 s. NN music

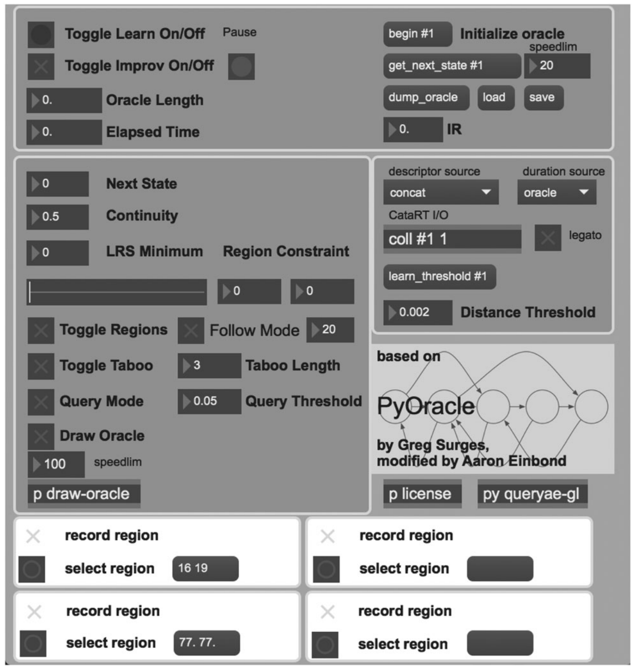

Figure 17. The interface of PyOracle.

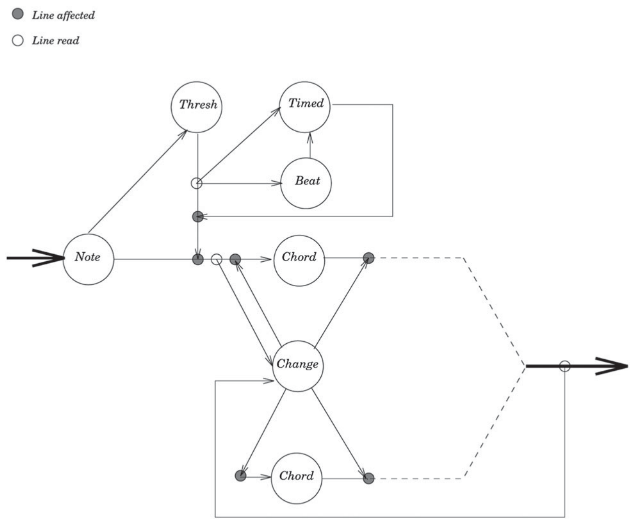

Figure 18. The Subsumption architecture of the Reactive Accompanist (Bryson, 1995).

includes two Multi-layer Perceptron (MLP) neural networks that are connected in series. Both networks are trained with the back propagation algorithm and have three hidden layers. The statistics of audio features is the input of the first ANN. The second MLP maps the classification output of the first MLP to synthesis parameters. The second MLP outputs a set of synthesis parameters with a probability distribution. Hence, the synthesis module inherits stochastic behaviours. The training of the first MLP is ongoing during the performance. There is a similarity algorithm in the system that checks if the current state is similar to the states that are used in the training. If the current state is not similar, ANNs are trained with the current state. The second MLP is trained before the performance.

Bown (2011) presents a musical agent $\textcircled{67}$ with continuous-time recurrent neural networks (CTRNNs). Each node is connected to each other with a directional weighted connection (synapses) in CTRNNs. In this implementation, the nodes have sigmoid activation function to process the directional weighted outputs of previous neurons. CTRNN is a blackbox type module with N (and M) floating point input (and output) values. In addition to the learning in CTRNN, Bown (2011) implements a mutation-only Genetic Algorithm (GA) that evolves multiple CTRNNs in parallel. Bown (2011) mentions that the GA includes a multi-objective fitness function that evaluates CTRNNs' 'success at acting with the responsive properties of dynamic reservoirs' and success at showing repetitive behaviours when the input is repetitive. This musical agent maps the CTRNN output to continuous synthesis parameters as well as the decision of triggering sound events. This agent has been presented in many concerts, performing with human-performers playing trombone, clarinet and shakuhachi. Building on this agent, Bown (2011) presents another musical agent that includes Decision Trees (DTs). Bown (2011) states five advantages of DTs over CTRNNs: discrete output on each time step, the ease in the analysis of the agent's behaviour, efficiency and adaptive self-calibration of decision boundaries. This implementation with DTs also includes a mutation-only GA evolving multiple DTs in parallel. The fitness function is single-objective, and it is for maximising the number of DT leaf nodes visited. This agent has also been presented in many venues, with human-performers playing trumpet, bass clarinet and electronics.

Kohonen Network is an application of ANN (Kohonen, 1998). Although Kohonen networks, including Self-Organising Maps (SOMs), are proposed in the early 1980s (Kohonen, 1982), it is recently discovered by studies related to the MuMe field. Smith Garnett (2012) present a musical agent $\textcircled{68}$ with adaptive resonance theory (ART) and reinforcement learning (the system number 69 in Table 1). This implementation focuses on monophonic melody generation. The ART network is a self-organising neural network for classification and categorisation of data vectors. ART network differs from SOM in training. Each input vector updates only one node in the ART network whereas in SOM, a set of nodes are updated. The agent converts the MIDI input to a combined feature vector of *pitch class, interval, interval and direction window, direction sign, octave* and *interval octaves*. The reinforcement learning implements two functions. First, when the agent updates an existing node in ART network, the agent calculates the reward using the previous and the updated state of the node. Second, when the agent creates a new node in ART network, the agent calculates the reward using a user set parameter called *vigilance*. The authors also present two examples of the agent on free improvisation. Another example presents the output of an agent trained using J. S. Bach's six unaccompanied cello suites.

Smith and Deal (2014) present a musical agent application $\textcircled{69}$ of SOMs. This agent architecture utilises chroma audio feature extraction in the perception stage to extract the pitch and rhythm information from the agent's audio input. Then, the agent's memory organises extracted chroma vectors in two levels, long-term and short-term. The authors introduce adaptive behaviour to the agent's short-term memory by using a SOM. In this system, training of the SOM is continuous. The decision module of this agent calculates a measure of learning in the SOM using the difference between the previous state and the trained state. Smith and Deal (2014) state that this learning measure is analogous to the Kolmogorov complexity. The decision module targets a learning rate. This agent follows its input if it is complex enough to satisfy the target learning rate. If not, the agent diverges from the input to increase the overall complexity. Hence, the decision module provides SOM a distance to the audio input vector. Then, the agent decides on a corresponding SOM node. This node is the input vector of the long-term memory. The long-term memory uses a *k–d* tree to search in a multi-dimensional space. Each vector provided from SOM is a search query to locate the closest vector in the long-term memory. The long-term memory consists of pre-defined audio files, and the agent does not update the long-term memory.

Martins and Miranda (2006) present a musical agent $\textcircled{70}$ with *SARDNET* generating rhythms. *SARDNET* is a variation of SOMs with an addition of temporality. *SARDNET* deals with event sequences using node activation values and differs from SOMs in two ways. First, the winning neurons are not included in the subsequent training. Second, the activation values of each node are decreased in each step. *SARDNET* represents the input sequence as all active nodes ordered by their activation values. Martins and Miranda (2006) approach the rhythmic events as three-dimensional events. These three dimensions are timbre, velocity and inter-onset interval. The musical agent includes two ANN cascaded in series. Symbolic representation rhythmic phrases (MIDI sequences) are the temporal input of the *SARDNET*. The output of *SARDNET* is connected to a one-layer Perceptron with three outputs. The training of this musical agent is through pre-recorded rhythms. The authors mention that after fifty iterations, the agent starts to self-organise. Notice that we also encountered the idea of evolving rhythms in Eigenfeldt's (Eigenfeldt, 2011) Kinectic Engine (see Section 5.1.2).

6.4. Hybrid musical agents with cognitive models

Camurri et al. (1995) present a musical MAS framework called *Hybrid Action Representation and Planning (HARP)* $\textcircled{71}$. Using the idea of graphical visual programming, *HARP* provides flexible programming environment to the user. The system is capable to create a hybrid agent system. The framework is inspired by MAS implementations in Robotics. The application of this framework is assisted composition, performance and analysis. The authors define two main components of *HARP*: *symbolic* and *sub-symbolic*. Symbolic components implement compositional syntax and semantics, including domain specific knowledge representations. Sub-symbolic components are the reactive modules of the system with a network of cooperative agents. Sub-symbolic components process the signals of MIDI, audio, or visuals. The authors also give an example of a theatre performance in which *HARP* framework is used to programme a software controlling sound, music and three-dimensional computer animation of humanoid figures interacting with real actors on stage.

The hybrid musical agent architecture $\textcircled{55}$ of Cont et al. (2007), mentioned in Section 6.1, explores musical agent applications using the mental representations of expectation in the problem of style imitation. There are four types of mental representations of expectation, proposed in the literature of psychology of musical expectation: verdical expectation (expectation of familiar works, related to episodic memory), schematic expectation (related to the semantic memory), dynamic adaptive expectation (related to the short-term memory), conscious expectation (related to the conscious reflection and prediction) (Huron, 2014). Cont et al. (2007) apply these ideas to MuMe by using an anticipatory model of musical style imitation with collaborative and competitive reinforcement learning. Within four types of anticipation (*Implicit, Payoff, Sensory* and *State*), the authors implement payoff and state anticipation models.

Similarly, Gifford Brown (2010) focus on the idea of using anticipatory timing to plan future actions of a musical agent. The system, called *Jambot* $\textcircled{72}$ is a hybrid musical agent that generates percussive musical rhythms. *Jambot* can generate rhythms by listening to other performers, or alone. The authors define anticipatory timing as a search for the best next note and when to play this note. The study stated that anticipatory timing enhanced greedy search while slightly increasing the computational complexity. Jambot includes a fitness function that evaluates possible actions and possible acting times for the next action. Jambot repeats the fitness evaluation on each time frame (audio frame). Gifford and Brown (2010) also present examples of system's output with and without anticipatory timing.

In the later versions of Jambot, Gifford (2013) introduces musical expectation in their hybrid musical agent. Jambot's application is percussive accompaniment to a live audio input. The authors are inspired by the previous works on musical expectation and propose metre as a framework for musical expectation. The system design involves metrical ambiguity to balance novelty and coherence. Jambot's architecture has three modes that controls level of metric ambiguity: disambiguation (use only the most plausible meter), ambiguation (use all plausible meters with equal weights) and following (use all plausible meters with the weights adjusted by plausibility). The reactive behaviours in this system include three approaches to fluctuate between imitative and intelligent actions: '(i) mode switching based on the confidence of understanding, (ii) filtering and elaborations of imitative actions, (iii) measured deviation from imitative action according to a salient parametrisation of the action space'.

Another recurrent theme in cognitive musical agents is motivation-driven, goal-oriented musical agent architectures (Beyls, 2008, 2009; Lynch, 2014). Beyls (2008, 2009) focuses on the motivation-driven musical agents $\textcircled{73}$. The architecture includes two-dimensional space (stability versus introverted-extroverted) to model behavioural changes. The system abstracts motivations as two types of drives: *integration* and *expression*. Integration drives aim to follow the input data whereas expression drives seek to move away from the input data. The *compound function* sets the drive of the agent depending on the levels of integration and expression. This hybrid musical agent implementation also includes reactive modules with an implementation of reinforcement learning and a Genetic Algorithm module (see 5.1.2) that evolves drives. Beyls (2009) also analysed the agent and shows that the fitness function of GA successfully follows the drives set by the compound function.

Lynch (2014) work is the only study that uses a cognitive architecture (CLARION) presented in the Cognitive Science. The system, called *Mocking-bird* $\textcircled{74}$ combines Van Nort's FILTER system with the *Clarion* cognitive architecture (Figure 19). The *Clarion* cognitive architecture consists of four sub-systems, that are the *Action Control System (ACS), Non-Action Control System (NACS), Meta-cognition System (MCS)* and *Motivation System (MS)*. Users can implement either the complete Clarion architecture or any number of its sub-systems. The *Clarion* architecture decides the actions of *Mocking-bird*.

These actions indicate a pre-recorded sample to be played starting from a point with a duration, and post processing effects such as pitch shift and time stretch.

The last four systems that we survey includes Affective Computing. First, *MAgentA* (75) is a cognitive musical agent that focuses on generating 'film like music' for games using an algorithm database with affective labels (Casella & Paiva, 2001). *MAgentA* is a part of the game framework *FantasyA* in which the user can influence the affective state of the characters they play (Paiva et al., 2002). *MAgentA*'s architecture has three modules: *perception, reasoning,* and *action*. This architecture resembles (Blackwell et al., 2012) PQf musical agent framework. Perception module checks the affective state of the environment and generates outputs when the affective state changes. Reasoning module checks if the new affective state can be generated with one of the algorithms in the database. If not, the exception handling module uses the history database to decide the most appropriate algorithm to use. Once the agent decides which algorithm to use, it sends the algorithm to the composition engine. The action module generates the audio output using data coming from the composition engine.

Second, Dubnov and Assayag (2005) combine the flow model with Factor Oracle and create a musical agent (76) within the *OMAX* framework. The agent listens to other performers online to train the FO. The flow model defines the notion of Experience Flow that explores the relationship of mental states with the activity where a subject is fully engaged and immersed with the tasks (Csikszentmihalyi, 2008). Dubnov and Assayag (2005) change the original flow model dimensions, *challenge* and *skill*, with two dimensions of *emotional* and *familiar*, and eight categories of *arousal, flow, control, boredom, relax, apathy, worry* and *anxiety*. The authors mapped these two dimensions of flow model variation to the *replication*, *innovation* and *recombination* parameters of their musical agents. These parameters controls the probabilities of links within the FO.

Third, Kirke and Miranda (2015) implemented Affective Computing with a virtual ecosystem. We mention other ecosystemic approaches in musical agents in Section 5.2.2. The application of the system is melody generation for assisted composition. The system is called *Multi-agent Affective Social Composition System* (*MASC* (77)) and combines Affective Computing with a MAS. The application in focus is assisted composition. MASC generates melodies through communication and artificial emotional influence between agents. This system implements affect estimation of musical melodies with continuous two-dimensional affective space. The dimensions are *valence* and *arousal*. This 2D model is common in Affective Computing in sound and music (Eerola & Vuoskoski, 2013). The agents situated in a virtual environment. The number of agents is in the range of 2–16. Each agent has a monophonic melody. The agents share their melodies with each other. Agents learn other agents melodies if the emotional state of the melody is close to the agent's emotional state. Moreover, the emotional states of agents are also affected by emotional states of other agents during communication. The authors present examples of melodies generated by this system. Also, the first author shared his compositions in which the first author used this system to generate melodies to assist the composition process.

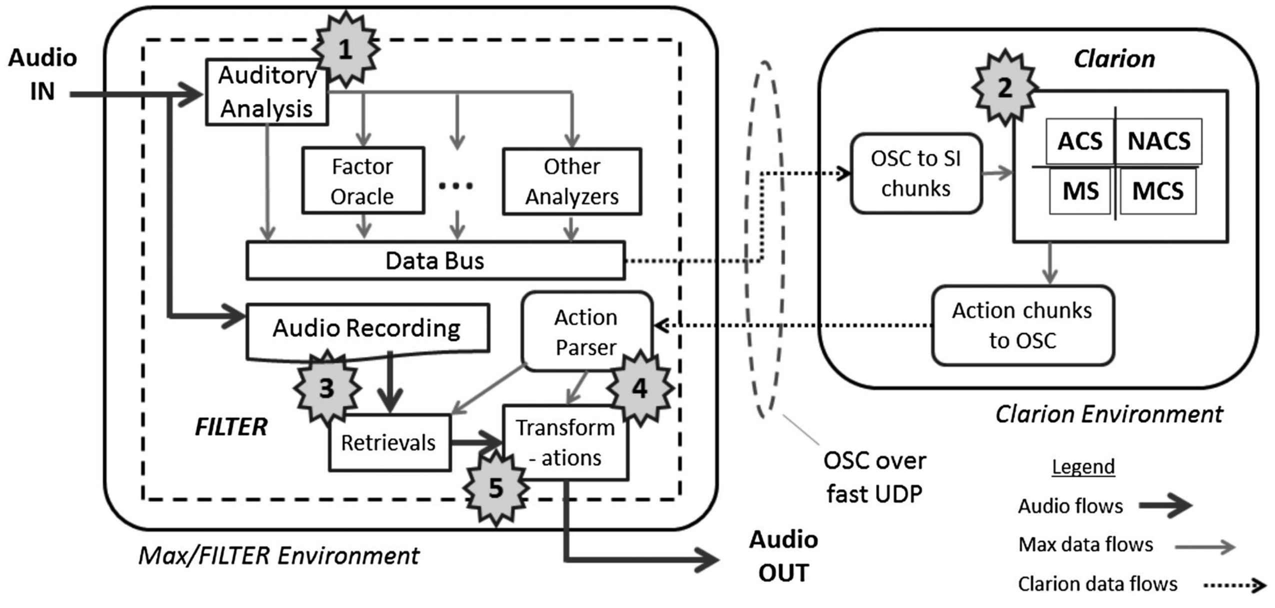

Figure 19. The architecture of Mocking-bird (Lynch, 2014).

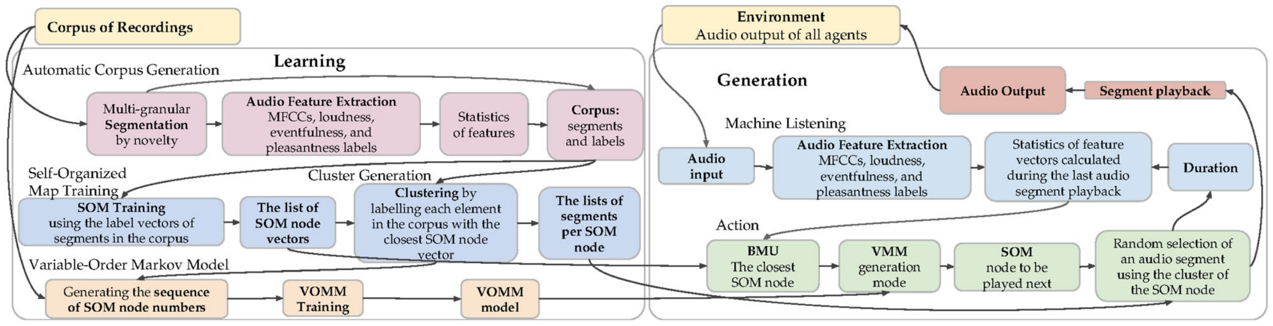
Corpus of Recordings
Learning
Automatic Corpus Generation
Multi-granular Segmentation by novelty
Audio Feature Extraction MFCCs, loudness, eventfulness, and pleasantness labels
Statistics of features
Corpus: segments and labels
Self-Organized Map Training
Cluster Generation
SOM Training using the label vectors of segments in the corpus
The list of SOM node vectors
Clustering by labelling each element in the corpus with the closest SOM node vector
The lists of segments per SOM node
Variable-Order Markov Model
Generating the sequence of SOM node numbers
VOMM Training
VOMM model
Environment Audio output of all agents
Generation
Audio Output
Segment playback
Machine Listening
Audio input
Audio Feature Extraction MFCCs, loudness, eventfulness, and pleasantness labels
Statistics of feature vectors calculated during the last audio segment playback
Duration
Action
BMU The closest SOM node
VMM generation mode
SOM node to be played next
Random selection of an audio segment using the cluster of the SOM node

Figure 20. The system architecture of *MASOM* (Tatar & Pasquier, 2017).

The last system that applies Affective Computing is *Musical Agent based on Self-Organising Maps* (*MASOM*) (Tatar & Pasquier, 2017; Tatar, Pasquier, & Siu, 2018). *MASOM* is a machine improvisation architecture for live performance (Figure 20). The musical context of *MASOM* is experimental music and free improvisation. *MASOM* is a flexible agent that can be trained on any audio file such as a recording of a performance or composition. *MASOM* extracts the musical form of an audio file using unsupersived learning. The learning stage has four steps. First, *MASOM* segments the audio file using the multi-granular segmentation (Lartillot, Cereghetti, Eliard, & Grandjean, 2013). Multi-granular segmentation uses novelty curve to segment an audio file. Second, each audio segment is labelled with duration, eventfulness, pleasantness and timbre features. Third, *MASOM* uses SOM to cluster these audio segments. The last step of the learning stage is the VMM training. Each segment of the original audio file is labelled with the closest SOM vector in the feature space. Using the order of segments in the original audio file, *MASOM* generates a string of SOM nodes. This string represents the musical form of the original audio. VMM is trained using this string of SOM nodes. The generation stage in *MASOM* includes online machine listening. The agent can listen to itself and other performers by extracting eventfulness, pleasantness and timbre features. MASOM uses machine listening module with the trained VMM to decide what to play next.

7. Evaluation of musical agents

Frayling (1994) proposes three types of research in Art and Design. First, the research into art and design is the historical, aesthetic, and perceptual research such as the Music History research. Second, the research through art and design includes the research of materials, customisation of technology, or procedures and results of practical experiments. Third, the research for art and design communicates the results of research through the end product, that is the work of art. The ideas and results are embodied in the artefact; hence, the verbal communication of the results is not the primary goal of this third type of research in art and design.

The developers of MuMe systems evaluate their implementations informally as a part of the software development. Hence, there are two classes of evaluation of MuMe systems: informal evaluations and formal evaluations. We differentiate the evaluation types of MuMe systems and musical agents with the following typology:

- **Informal Evaluations** does not involve formalised research methodologies.
  - **The authors** are the creators of MuMe systems.
  - **Users, peers and experts** are the close entourage of authors.
  - **The audience** is the recipients of the artworks generated by the MuMe systems.
  - **The media** covers critical writings of experts in art and music.
- **Formal Evaluations** are formalised methodologies to assess the success of MuMe systems.
  - **Peer reviewers, curators and jury** give direct and indirect feedback to the authors of MuMe systems.
  - **Theoretical and analytic measures** are the formal evaluation methodologies that does not involve human participants. These methodologies are synthetic measurements that are acknowledged in the academia.
  - **Empirical studies** apply quantitative, qualitative and mixed methodologies with human participants.

7.1. *Informal evaluations*

Informal evaluations do not involve any established research methodology. Informal evaluations of musical agents start at the beginning of system's ideation. The authors iterate the architecture and the system parameters as a part of the development process. This process includes many iterations in which the authors evaluate the system's output, change the parameters or the agent architecture, and evaluate the system's output again. For example, most machine learning algorithms require a set of parameters to be decided by the developer. These parameters are mostly set by many trials and errors with the system.

Colleagues and friends of authors are the subjects of another type of informal evaluations. The authors have a different perspective on the system with the feedback of people who are close to the authors. When the system's output is publicly shared, the authors receive feedback from the audience. Although this type of feedback is still informal, it is beneficial to evaluate the initial results of the system. When the system outputs reach the media such as journalists, critics, software testers, bloggers; the authors receive a feedback about social implications of systems.

7.2. Formal evaluations

Formal evaluations use established research methodologies to answer a clearly defined research question to assess a system (Arges, Forth, & Wiggins, 2016) two types of formal evaluations: internal and external.

Internal formal evaluations are conducted during the generation stage of a musical agent. Musical agents assess their creative output during the generation to improve the output. In most cases, musical agent developers implement the internal evaluation as system feedback loops or using agents with evaluation roles. For example, Cypher (10) uses its listener agents to evaluate player agents. The internal evaluation is a part of system design, and we already covered the system design of musical agents in the previous sections.

External formal evaluations are conducted after the agent finishes its performance or generation. There are four aspects of external formal evaluations:

- **Dimensions of evaluation:** Three types of evaluation dimensions are common in the CC literature: software validation, the quality of a system's output and creativity (Ritchie, 2014). In MAS, the authors can study the creativity of one agent or the creativity of the system output. Most synthetic evaluations research the effect of hyper-parameters on the system output. A hyper-parameter is a common term in Machine Learning and it refers to the parameters of system design, such as number of agents in MAS, or genetic operator probabilities in EC. In most cases, hyper-parameters are set before the system run.
- **Participants of evaluation:** The authors develop MuMe systems to be used by a user to generate music that is presented to an audience. Therefore, researchers focus on three types of participants in the evaluation: the authors, the users and the audience. Researchers can evaluate the research and development process of authors, the interaction of users with the finalised system, and the audience response to the output generated by the system.
- **Output selection:** Ritchie (2014) proposes five types of output selection that appears in CC: recreating known exemplars of the domain, exploring the neighbourhood of these exemplars, exploring the parameter space of a system, random sampling of the parameter space, structured sampling of the parameter space. These selection options are also applicable to MuMe systems.
- **Methodology**: Software validations, synthetic evaluations and empirical evaluations are the tools of formal evaluations.

7.2.1. Software validations

The musical agents that we mention in this survey are written as software codes. Software evaluations use software validation techniques in Computer Science to assess if the code implementation is sound, complete, or stable. Briefly, the following three techniques come forward in Computer Science:

- **Formal Validations**: Mathematical proofs of the system behaviour are examples of this type of validations.
- **Black Box Tests**: Given pre-set inputs, black box tests study the system output to evaluate a system's behaviour.
- **White Box Tests**: Given a set of selected inputs, white box tests exhaust a system for all possible conditions to ensure stability and robustness.

7.2.2. Synthetic evaluations

Synthetic methods do not involve human participants. Theoretical, analytical and computational tools are the methods of synthetic evaluations. Because of the particularities of musical agent implementations, the authors create new synthetic methodologies that are specific to their implementation. In the following, we survey synthetic evaluations of *Beatbender* (20), system (32), *VMMAS* (1), *IMAP* (36) and *pMIMACS* (46).

Levisohn and Pasquier (2008) evaluated *Beatbender's* (20) system output using two criteria: emergence and complexity. The authors assessed the emergent behaviours of the system by analysing interaction between agents, and the complexity by comparing subsequent patterns generated by the system. The authors reported that *BeatBender* ㉒ successfully generated emergent rhythms by taking advantage of the Subsumption architecture.

Aucouturier (2011) also evaluated the emergence and convergence in the multi-agent society ㉜ that evolves tuning systems. The author evaluated the system with different agent interaction types: single note shift interaction with harmonic timbre, drone shift interaction with harmonic timbre, drone shift interaction with compressed timbre, and drone shift interaction with a society of agents with harmonic and compressed timbre. Hence, the author researched the effect of agent interaction types on the system output. Aucouturier (2011) concluded that the system could emerge coherent tuning systems through local agent interactions in the multi-agent society.

The following three evaluations studied the effect of hyper-parameters on systems' output. Vicari et al. (2005) evaluated *VMMAS* ① with two evaluations. Both evaluations calculated a variable called *synchronism property*, which is calculated using the rhythm generated by the agents. However, the authors concealed the details of how to calculate this parameter. The authors claimed that synchronism property above 60 indicates a 'good performance'. The first evaluation included only software agents, and concluded that introducing new agents to *VMMAS* ① influenced the overall synchronism. Hence, the authors studied the effect of a hyper-parameter, that is the number of agents. We mention the second evaluation of *VMMAS* in Section 7.2.3.1.

Miranda et al. (2010) conducted three evaluations to evaluate IMAP's ㊱ performance. The first evaluation studied if agents could perform according to their individual preferences. The weights of the rules in an agent's fitness function indicate the individual preferences of an agent. This evaluation concluded that average agent performances were correlated with the preferences of agents. The second evaluation showed that the user can control the overall diversity of performances by changing the spread of the rules weights. The third evaluation researched if the population was affected when a subset of agents in the population are biased in their fitness function. This third evaluation showed that IMAP ㊱ could direct the performance diversity to a region in the search space by introducing a bias to a subset of the population.

Kirke Miranda (2011) analysed *pMIMACS* ㊻ output with a synthetic evaluation with three agents. This evaluation was the detailed analysis of two runs of MAS. The first run was 8 episode long whereas the second one was 10. During each episode, agents with performer roles performed for the agents with listener roles. For this evaluation, the agents were initiated with a unique melody including four notes. All notes were sixteenth notes generated by random walk. The authors claimed that *pMIMACS*'s outputs were less mechanical than the outputs that were 'usually produced by the algorithmic compositions systems'. However, the study did not include any empirical evaluation to support this claim.

None of the synthetic evaluations study the creativity of musical agents. However, musical agents tackle musical creative tasks, and the assessment of creativity is crucial to evaluate the success of a system.

7.2.3. Empirical evaluations

Given that the definition of creativity is still in discussion (Peter, 2009), empirical evaluation methodologies handle the complexity of creativity assessment by using human participants to judge the output of a system. Before going into the details of these evaluations, let's cover the background of creativity.

Boden (2015) defines creativity as 'the ability to generate new forms'. This definition explains creativity by focusing on the artefact. Boden (2015) continues by proposing *psychological* and *biological* creativity to categorise human and non-human creativity. *Biological* creativity is 'the ability to generate new cells, organs, organisms, or species'. We explored computational abstractions of biological creativity in musical agents in Section 5.2. In comparison, *psychological* creativity is 'the ability to generate ideas and/or artefacts that are new, surprising, and valuable'.

Boden (2015) focuses on two key points to understand which forms are new. First, Boden (2015) discusses the notion of *novelty* in creativity. Second, *historical creativity* is a special case of psychological creativity in which generated form is novel to the community. Furthermore, Boden (2015) states three types of creativity as a result of *mechanisms* generating novelty; *exploratory*, *combinational* and *transformational*. First, *exploratory* creativity is making novel forms that satisfy constraints of a particular style. An example of exploratory creativity is improvising a Jazz melody in the style of Charlie Parker. Second, *combinational* creativity is combining styles in novel ways such as improvising a Jazz melody in the style of Chet Baker with the ornamentations of Charlie Parker. Third, *transformational* creativity is the expansion of known conceptual space. An example of transformational creativity is John Cage's idea of including random sounds of audience to a musical performance.

How to assess the creativity of a system has been a challenge for the CC research. Jordanous (2012) pointed out the lack of evaluation in the publications of CC systems. Also, within the publications that evaluated their systems, the evaluation of creativity was not common. When the creativity evaluation took place, the participants were mostly the people who implemented the system. Jordanous (2012) emphasised the lack of evaluation criteria in the publications that evaluated creativity. According to 2012, there was a clear lack of connection between the evaluation of CC systems and the evaluation methodologies that were presented in CC. Currently, there is still no evaluation methodology that is accepted as a standard in CC. Jordanous (2012) also stated that CC inclined towards the evaluation of the quality in comparison to the evaluation of creativity. We observed similar tendencies in the evaluation of musical agents.

There has been recent attempts to categorise the evaluation methodologies for MuMe systems. Arges et al. (2016) identified six types of external evaluations:

- Behavioural Tests
- Consensual Assessment Technique (CAT)
- Extensions within Computational Creativity
- Questionnaires, Correlational Studies and Rating Scales
- Physiological measurements and neurophysiological measurements

Regarding the evaluation of creativity, we observed two common cases in the empirical evaluations of musical agents. In the first case, the authors evaluated the systems from the user perspective. The participants were expert users who tried the musical agent. Although these evaluations did not necessarily follow the typical CAT methodologies, we grouped them under the CAT category since the participants were expert users. In the second case, the authors evaluated a musical agent from the perspective of the audience. In these evaluations, the evaluation tools were questionnaires and rating scales. Lastly, we observed only one system that incorporated an evaluation methodology from CC. In the following, we go into details of musical agent system evaluations on creativity.

7.2.3.1. Consensual assessment technique. A group of experts evaluate the creativity of MuMe systems in Consensual Assessment Technique (CAT). In musical agents, the expert evaluation refers to quantitative and qualitative empirical evaluations with expert participants, and case studies. For example, the second evaluation of VMMAS ① (Vicari et al., 2005) studied a performance session with software and human agents. The human performer reported that the system was successfully accompanied with satisfactory rhythmic and harmonic behaviour.

Navarro et al. (2016) presented an example of mixed method empirical evaluation study with *MUSIC-MAS* ⑥ that assists composers by generating harmony progressions. The participants were novice composers who were studying first year music theory at university. The authors asked the participants to compose their first piece using a harmony progression generated by *MUSIC-MAS*. The participants rated each others' compositions as well as *MUSIC-MAS*' success on assisted composition. This evaluation concluded that MUSIC-MAS could help novice composers by assisting composition tasks.

Murray-Rust and Smaill (2011) carried out several case studies to evaluate their musical agent based on *MAMA* ⑨. The case studies were a duo of a human performer and

- another human performer
- a recording of a human performer
- a musical agent without expressivity and interactivity
- a musical agent mirroring the human input
- a musical agent including Musical Acts

However, this empirical evaluation concluded that the introduction of Musical Acts did not significantly change interactivity, competence and expressivity, and general performance.

Linson et al. (2015) conducted two qualitative evaluations with Odessa ⑯ that is a mono-agent system including machine listening. The first study included eight expert musicians playing clarinet, trumpet, cello, soprano saxophone, guitar, bassoon, piano and vocals. Six of eight musicians reported 'a process of familiarisation and improved collaborative engagement' while two musicians were highly dissatisfied. After the first evaluation, the authors added a module of excitation to the architecture so that the system responds to higher activities in the input. The second qualitative evaluation had two expert musicians playing soprano saxophone and guitar. These musicians also participated in the first evaluation. The evaluation was a trio session including Odessa. The musicians reported a 'coherent identity' of Odessa regarding both evaluations.

Collins (2011) evaluated *LL* ㉕ with two expert musicians. One of the experts was a percussionist whereas the other one was a violinist. Both sessions were presented as public concerts. The percussionist conceptualised *LL* as the extensions of its programmer. The violinist mentioned the trade-off between the controllability versus agency of the system.

Similarly, Aucouturier and Pachet (2005) pointed out the trade-off between autonomy and reactivity in the evaluation of Ringomatic $\textcircled{52}$. The evaluation was a case study of *Ringomatic*'s interaction with a human drum player. The authors clarified that *Ringomatic* could follow the human performer while preserving the global continuity.

Eigenfeldt and Pasquier (2011a) carried out a quantitative listening evaluation to evaluate *Coming Together: Free-sound* $\textcircled{25}$. Four soundscape compositions were generated for the evaluation. One composition was generated by the system. Another one was generated by random. The remaining two were composed by an expert composer. One of the human composed ones was freely composed without constraints whereas the second one was limited with database, methods of processing, overall duration, static spatial distribution of four gestures in four channels. The evaluation survey questions focused on the soundscape characteristics, compositional success, skill level and subjective reaction. In all cases, the system was better than the random generation.

Hawryshkewich et al. (2010) carried out a case study with beginner drum players to test if *Beatback* $\textcircled{51}$ could improve the self-directed learning of drum players. The authors reported that 'the majority of participants felt less enjoyment and more tension with drum zoning enabled.'

Surges and Dubnov (2013) tested the capabilities of *PyOracle* $\textcircled{59}$ with a case study. *PyOracle* $\textcircled{59}$ was presented in a public concert as a performance with an expert musician. The case study was a *structured improvisation*. Structured improvisation is free improvisation with predefined constraints for musical sections. The case study was a concert performance including a score for both *PyOracle* and the human performer. The performance was followed up with an interview with the human performer. The performer emphasised that the flexibility of *PyOracle*'s timing mechanism could be elaborated.

Sampaio et al. (2008) presented two empirical evaluations assessing the quality and diversity of *CinBalada*'s $\textcircled{64}$ musical output. The first evaluation showed that the participants preferred *CinBalada*'s output over random generated or similarity-based rhythms. The second evaluation concluded that the participants found the diversity of *CinBalada*'s output not too distant from the diversity of randomly generated rhythms.

In these evaluations, we observe that the details of methodology are not clear and the justification of the proposed methodology is missing. The main discussions around the formalisation of expert studies are still to be done in MuMe. Notice that, the evaluation of *Odessa* $\textcircled{16}$, *LL* $\textcircled{25}$ and *PyOracle* $\textcircled{59}$ were conducted through post-performance interviews. These interviews did not follow a typical qualitative methodology. Regarding all CAT evaluations, the hypothesis is not clear and the dimensions of evaluation are vague. The evaluations are exploratory; however, this fact is implicit and there is no justification of why an exploratory approach is chosen.

7.2.3.2. Evaluation methodologies of Computational Creativity. This type of empirical evaluations integrate the methodologies of CC to evaluate MuMe systems. Many evaluation methodologies have been proposed in CC, such as Standardised Procedure for Evaluating Creative Systems (SPECS) (Jordanous, 2012), the Creative Tripod (Colton, 2008) and FACE/IDEA model (Pease & Colton, 2011). We have found only one study that incorporated a methodology from CC to evaluate musical agents. Yee-King and d'Inverno (2016) used MusicCircle, a timeline-based tagging and annotation system to evaluate *Speake-System* $\textcircled{62}$. The conclusion of the qualitative study was that the system gave a strong sense of interaction; however, failed to generate long-term structures. By no means this survey covers all discussions around the assessment and evaluation of creativity and proposed evaluation methodologies in CC. Still, only one study incorporated a CC methodology to evaluate a musical agent. Hence, this creates opportunities to integrate CC evaluation frameworks to musical agents.

7.2.3.3. Questionnaires, correlational studies and rating scales. Surveys and questionnaires are one of the main tools that musical agent developers use to evaluate their applications. In comparison to CAT, the participant group is not a group of experts in this type of evaluations. We have found three systems with such evaluations.

Murray-Rust et al. (2005) conducted a questionnaire that is similar to the Turing Test (Turing, 1950) to evaluate their rhythm generating system *VirtuaLatin* $\textcircled{18}$. Turing test is one of the first methodologies that is proposed to evaluate automatic agents. Murray-Rust et al. (2005) concluded that the general public could not differentiate the machine-generated rhythm from a human-generated one while a higher percentage of expert listeners could.

Delgado et al. (2009) evaluated *Inmamusys* $\textcircled{2}$ by generating four compositions with the input affective labels *worry, happiness, chaos* and *worry* again. The participants labelled these compositions with affective states of *sadness, happiness, fear, worry, chaos,* and *indifference*. The authors reported that the participants affective labels were in line with the input affect labels of the generated compositions.

Kirke and Miranda (2015) conducted a listening test with ten participants to evaluate *MASC* $\textcircled{77}$. The evaluation aimed to see if the affective states of single agents could be observed in the melody output of the Multi-agent system. The results indicated that the affective states of single agents appeared in the final output. Still, the authors mentioned that a following this first evaluation with another one with more participants is required to conclude on a significant result.

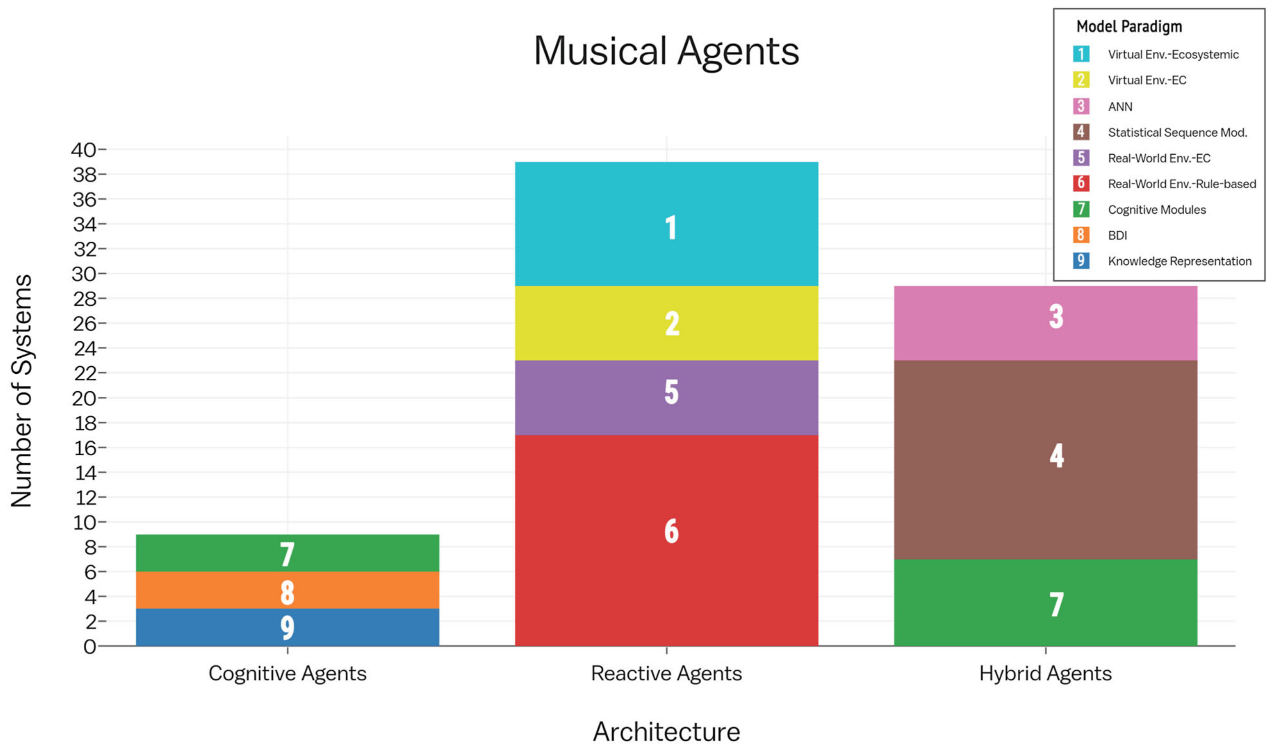

Figure 21. The number of musical agents per architecture type.

7.3. Future steps of evaluation and benchmarking

We observe that the evaluations of musical agents apply system specific methodologies. The dimensions of evaluations are not clear in most cases and the justification of why a particular dimension of a system is evaluated is missing. Given that the hypothesis and the methodologies of evaluations vary, no benchmarking tasks were initiated for musical agents. An obstacle for benchmarking is the reusability and code availability that we mention in Section 8.3.

The MIR field has developed a set of benchmarking tasks and through the Music Information Retrieval Evaluation eXchange (MIREX), the MIR field addresses formally defined challenges. Musical agent researchers could apply a similar approach for benchmarking. For example, many systems tackle style imitation tasks and it could be possible to benchmark these tasks. As of 2017, Institute Neukom have sent a call for Music Creative Turing Test 2018.[12] This is a recent benchmarking attempt for MuMe systems including musical agents.

Although we covered the musical agent evaluations that applied consensual assessment technique, evaluation criteria from Computational Creativity, questionnaires, correlational studies and rating scales; we found no study that applies behavioural tests, physiological and neurophysiological measurements. behavioural tests assess divergent thinking, convergent thinking, artistic ability and self assessment. Some examples of behavioural tests in Music are Measure of Musical Problem Solving (Vold, 1986) and Measure of Creative Thinking in Music II (Webster, 1987). Physiological and neurophysiological measurements analyse the physiological response of audience. Motion capture, eye tracking, galvanic skin response, Electroencephalography (EEG) are the examples of tools to measure audience physiology. It could be also possible to apply the measurement neural responses of audience to evaluate the performance of a musical agent. However, these technologies are particular to specific areas and applications, and they are not always available in the institutes that research MuMe.

8. Ad infinitum

8.1. Architectures and algorithms

Figure 21 presents the number of systems for each architecture type. We observe that the number of reactive musical agents is the highest, followed by the number of hybrid musical agents. EC modules appear both in reactive agents in real-world and virtual environments. Moreover, cognitive modules show up in cognitive and hybrid agents. However, the number of cognitive musical agents is the lowest.

We have found only 16 implementations of musical agents with cognitive modules, including the hybrid

[12] http://bregman.dartmouth.edu/turingtests/music2018

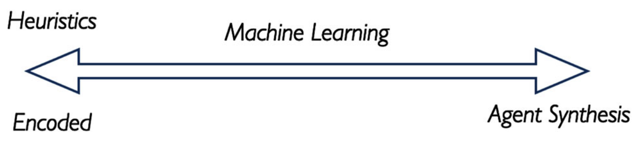

Figure 22. The continuum of autonomy in musical agent design.

musical agents with cognitive models (Figure 21). Thórisson and Helgasson (2012) present the state of the art cognitive architectures: *Ymir, ACT-R, Soar, NARS, OSCAR, AKIRA, CLARION, LIDA, Ikon Flux*. Except for *CLARION*, we have not encounter any study in which any of these architectures are applied to a MuMe task. Notice that, cognitive musical agent studies are challenging because applying a cognitive architecture to a musical task requires the expertise in Music, Computer Music, AI, MAS and Cognitive Science. Also, these cognitive architectures are reasoning architectures and they are not music cognition architectures. The research on Music Perception and Cognition is still to be reflected to the cognitive musical agent studies.

Regarding musical agent studies with statistical sequence modelling, there is still more to be done to generate variety in longer musical sections. Pachet (2003) and Assayag and Dubnov (2004) clarify that Markov Models fail to represent the conditional probabilities of sequences longer than the order. Hence, many of the systems presented in Section 6.1 do not include long-term memory and one can argue that these systems fail to produce variety in long-term musical sections and structures. Dubnov et al. (1998) and Pachet (2003) address this problem by introducing interactivity to Markov Models. Hence, the generation of long-term structures guided by a human performer. *Improtek* comes forward with the idea of using *scenario generation model*, and combining probabilistic methods with Factor Oracle is another promising approach to generate long-term continuity (Déguernel, Vincent, & Assayag, 2018).

ANN algorithms are still to be examined by the musical agent developers. With the increasing research on Deep Learning, a variety of new algorithms as well as improvement of the previous algorithms are presented in the literature (Arel, Rose, & Karnowski, 2010). Briot, Hadjeres, and Pachet (2017) surveyed Deep Learning approaches for musical tasks. Although these systems are mostly purely generative systems, it is possible to incorporate these approaches with MAS to develop musical agents. Moreover, we have found only one study (system (67) mentioned in Section 6.3) that evolves ANN modules using NeuroEvolution of Augmenting Topologies (NEAT) (Bown, 2011). NEAT combines ANN with EC to evolve ANN modules.

Genetic Programming (GP) is a type of EC algorithms. We have not found any musical agents applying GP in the system design. GP, especially Cartesian Genetic Programming (CGP), has been applied to image recognition (Harding, Leitner, & Schmidhuber, 2013) as well as style imitation in Visual Arts (Miller, 2011). Moreover, Wooldridge (2009) proposes the idea of synthesising agents. In all musical agent systems that we have covered, the authors develop the systems manually. Automatic musical agent design is possible using GP and CGP algorithms. GP and CGP have been applied to synthesise audio synthesis architectures (Allik, 2014; Garcia, 2001; Macret & Pasquier, 2014; Takala, Hahn, Gritz, Geigel, & Lee, 1993; Wehn, 1998). It is possible to improve the automatic audio synthesiser design systems to synthesise musical agents.

8.2. Interdisciplinarity of MuMe

While the International Workshops on Musical Metacreation[13] have covered MuMe topics for five MuMe workshops, five MuMe concerts, and three MuMe tutorials since 2012; the topics of MuMe field has been covered by various platforms such as International Computer Music Conference,[14] the International Symposium for Music Information Retrieval,[15] the Sound and Music Computing,[16] the Association for Computational Creativity,[17] the International Computer Music Association,[18] the International Conference on New Interfaces for Musical Expression,[19] Live Coding,[20] the International Symposium for the Electronics Arts,[21] the conferences held by the Association for the Advancement of Artificial Intelligence.[22]

In some cases, the success of MuMe systems are dependent on the advances in other disciplines. For example, many agents working with audio or hybrid I/O include machine listening. Examples of machine listening tasks are tempo estimation, fundamental pitch detection, sound similarity, rhythm similarity, melody similarity, affect estimation in sound, chord analysis, audio thumbnailing, novelty detection, etc. The MIR field addresses these tasks and many are still open research questions. By default, the success of musical agents with machine listening relies on the quality of the machine listening algorithm. Hence, these musical agents are dependent on the advances in MIR studies.

[13] http://musicalmetacreation.org/
[14] http://computermusic.org/
[15] http://www.ismir.net/
[16] http://smcnetwork.org/
[17] http://computationalcreativity.net/home/
[18] http://computermusic.org/
[19] http://www.nime.org/
[20] https://toplap.org/
[21] http://www.isea-web.org/
[22] https://www.aaai.org/

Therefore, the MIR and MuMe fields naturally benefit from each other by putting forward new problems and solutions. For example, MuMe uses advanced technologies of MIR in machine listening and automatic extraction of higher level music features. Also, musical agents can utilise the recent developments in Affective Computing in Sound and Music (Eerola & Vuoskoski, 2013; Fan, Tatar, Thorogood, & Pasquier, 2017) in machine listening modules of musical agents. Other MIR areas are also valuable to musical agents such as musically informed audio decomposition, tempo and beat tracking, chord recognition and music structure analysis (Müller, 2015). Likewise, MIR can take advantage of the MuMe research. For example, Collins (2017) proposed autonomous critic agents in the assessment of musical style, novelty, or quality. Collins (2017)'s study proposed a model for critic agents that have listened to more music than humans could. Such critic agents can be explored for the tasks of recommendation systems in MIR.

8.3. Design considerations

The developers of musical agents create a system architecture by going through a design process. Autonomy in musical agent design ranges from encoded systems to agent synthesis (Figure 22). The developers design the system architecture manually in the encoded and heuristics systems. In comparison, agent synthesis is completely autonomous (Wooldridge, 2009) and can generate musical agent architectures. We propose to refer to the phenomena of agent synthesis as Metacreation of Metacreation (Meta^{2}creation). We claim that Musical Meta^{2}creation is developing systems that create systems that partially or completely automatise musical tasks.

Machine Learning lies in the middle of the autonomy continuum in the musical agent design. The developers of musical agents often incorporate Machine Learning in their system design. Machine Learning algorithms have parameters to be set by the developers. For example, an EC algorithm has genetic operator probabilities that set the chance of applying the genetic operators to an individual. Another example is the highest order parameter in Variable Markov Models. The developers often set these parameters by listening to the system output for various parameter options. This process is addressed in the Machine Learning as Interactive Machine Learning or User-Centered Machine Learning (Bernardo, Zbyszynski, Fiebrink, & Grierson, 2016; Gillies et al., 2016). The research on the procedures, tendencies and underlying factors of developing musical agents is still to be done.

We have encountered three recent studies that studied the design principles of Computational Creativity (CC) systems including musical agent systems (Bray and Bown, 2014, 2016; Bray, Bown, & Carey, 2017). First, Bray and Bown (2014) compare user experience of a DAW and the musical agent Nodal (42). Second, Bray and Bown (2016) propose applying the Interaction Design theory to CC systems. Third, Bray et al. (2017) compared three generative music systems to understand the effect of the degree of encapsulation in MuMe systems. The study included a direct manipulation system, a programmable interface system and a highly encapsulated system.

Reusability and code availability is another issue of musical agents. Out of 78 systems, the source codes of 18 systems (23%) are available to the public (Table 1). This makes the comparison of different musical agents difficult. Addressing this issue, the manifesto of Musebot framework encourages making musical agents open-source by publicly sharing the code of the framework and submitted musical agents (Bown, Carey, & Eigenfeldt, 2015). *Musebot* project is a framework for musical agents which allows interactive live performances with human performers and multiple musical agents (Eigenfeldt, 2016b; Eigenfeldt, Bown, & Carey, 2015). The system design of the framework is the client/server architecture in MAS. As of 2017, Musebot framework is compatible with MAX, Max for Live, PureData, Processing, SuperCollider, Python, Extempore and JAVA. The framework provides exciting opportunities such as collaborative performances of various musical agents and autonomous curation of musical agent ensembles. The Musebot framework provides an opportunity to create a public repository of musical agents.

Table 1 shows which systems have been presented to the public within our knowledge. We also include systems implementing assisted composition tasks, if the systems have been used to produce a composition that is presented to the public. Out of 78 system, 39 systems (50%) have been presented in public venues. Musical agents can aim for increasing the percentage of systems available to the public.

8.4. MuMefication

8.4.1. MuMe as a field

There is an objective evidence that MuMe is an interdisciplinary field. In a recent paper devoted to this topic, Bodily and Ventura (2018) clarify that total 80 papers were published in the 5 MuMe Workshops between 2012 and 2017. These papers had a total 111 authors. Out of these 111 authors, 88 (79.2%) published only once, 13 (11.7%) published twice and 8 (7.2%) published three or more times in MuMe Workshops. Out of these 80 papers, 36 of them had 173 external citations in total. In comparison, there were only 13 instances where MuMe papers cited other MuMe papers. The higher rate of external citations in comparison to internal citations of MuMe publications, and low re-publication rate of authors indicate that MuMe is a growing interdisciplinary field. (Bodily & Ventura, 2018) mention that the external citations of MuMe papers appeared in papers presented in a variety of venues such as the International Conference on Computational Creativity (ICCC); Computers in Entertainment (CIE); the Computer Music Journal (CMJ); the Sound and Music Computing Conference (SMC); and the International Computer Music Conference (ICMC).

We propose that MuMe as an interdisciplinary field inherits both practice of generative music whether it is artistic, heuristic, creative AI; as well as the scientific study of Computational Creativity for musical creative tasks. We think that the term MuMe can provide a formalisation of such systems using the definitions and ideas of Computational Creativity, Generative Art and Artificial Intelligence. We propose that MuMe as an interdisciplinary field aims to bring together all fields that apply autonomous approaches for creative musical tasks. Our definition of MuMe as a field is inclusive in the sense that it is not creative musical AI because it is not always AI, it is not a simulation of musical creativity because MuMe can also cover *creativity as it could be*, it is not necessarily live-coding because MuMe systems are not necessarily performed live, it is not artificial life because it also covers systems that do not simulate a virtual environment. In addition, we think that CC literature gives an established ground to explain whether MuMe systems are musically creative, and if they are, the kind of creativity that MuMe systems output.

8.4.2. A typology of MuMe systems

The discussion around the typology of MuMe systems is still ongoing. Eigenfeldt, Bown, Pasquier, and Martin (2013) propose a taxonomy of MuMe systems in seven levels of independence, compositionality, generativity, proactivity, adaptability, versatility and volition; ordered from least autonomous to the most. Based on these seven levels of MuMe systems, we propose six levels of musical agents which are ordered from the lowest level to the highest one:

(1) Reactivity: Agents respond to the changes in the environment in a timely fashion.
(2) Proactivity: Agents can perceive their environment and plan future actions.
(3) Interactivity: Agent can interact with other agents (human, artificial, or biological).
(4) Adaptability: Agents learn from their environment to improve competence or efficiency.
(5) Versatility: Agents are domain independent.
(6) Volition and framing: Agents can explain why they choose certain actions when asked by other agents.

The higher levels can inherit properties of the lower levels whereas the lower levels cannot present the distinctive properties of the higher levels. Many agents that we cover demonstrate reactivity behaviours. For example, *Odessa* (16) can react to the musical actions of other performers. Odessa also exhibits interactivity by influencing other agents by actions. *Odessa* diverges from the current state of the environment if other agents fail to generate variability. However, *Odessa* does not learn from the environment. In comparison, system (69) exhibits adaptivity by training the SOM in the architecture online. The musical agents that are free of the author's style or choice show behaviours of Versatility. For example, the *Continuator* (50) is a flexible agent that imitates the style of any performer.

Although the author's style is not explicitly embedded in the *Continuator*, one can argue that by just choosing one generative algorithm over the other, the authors make implicit stylistic choices on the design of the musical agent. Thomas et al. (2013) studied the bias of three style imitation algorithms in melody generation. The authors compared VMM, FO and *MusiCOG* (8) and concluded that each algorithm introduced a particular bias to the melody generation. One can argue that the *Continuator* is not completely independent of the author's style because the author made the decision on the generative algorithm that was used in the system design. Hence, the selection of one melody generation algorithm rather than others introduced a particular bias to the Continuator.

The taxonomy of Eigenfeldt et al. (2013) distinguish MuMe systems based on the dimension of autonomy. In comparison, Blackwell et al. (2012) propose a taxonomy of MuMe systems by focusing on the system architecture. The authors propose a new term: live algorithms. Live algorithms include musical agents as well as purely generative music systems that do not utilise any input in the system design. The authors clarify four main interaction types of Live Algorithms: autonomy, novelty, participation and leadership. The authors present eight case scenarios of system designs. Each case has a different combination of incoming audio stream, outgoing audio stream, human control and three modules of P (listening/analysis), Q (performing/synthesis) and f (patterning, reasoning, or intuition). Moreover, the authors state four types of Live Algorithm behaviours: shadowing, mirroring, coupling and negotiation. The authors continue by presenting implementations of Live Algorithms and further considerations.

8.5. Challenges and opportunities

There are several reasons why the research and development of generative systems and musical agents matter. The main usage of computational systems has shifted from rational problem solving. With the increasing number of personal computational systems, the percentage of computational power that is used for entertainment, art and culture increased rapidly.

As a result, the demand for generative systems including musical agents in the creative industries escalated. This demand arises from the growth of non-linear media. Non-linear media enable users to choose from the available options in the media. Hence, non-linear media are interactive by nature. Two examples of non-linear media are games and websites. The workload to generate content for non-linear media is vastly greater than the workload of linear media production. Hence, there is an increasing demand for autonomous, adaptive systems that can fulfil the requirements of non-linear content. In that sense, we can use musical agents in the industry of non-linear media as adaptive and autonomous systems making music. Moreover, these autonomous systems can enable the personalisation of the content. That is, the software can adapt to the user's specific choices, aesthetics and requirements.

MAS are applied to simulate real-world phenomena. We can apply musical agents to simulate and study musical phenomena. Using such simulations, we can model and study (software or human) agent interactions and emergent behaviours in musical tasks. In MuMe, this is referred as modelling creativity as it is (Pasquier et al., 2017). These simulations can help understanding how we make music.

In comparison to musical creativity as it is, musical agents introduce new opportunities for the exploration of musical creativity as it could be. One advantage of software agents is that agents can both play music, listen and exchange messages about their beliefs, desires and intentions during a performance. The rate of communication can be much higher than that of human communication. Also, software agents can be shared easily over the internet and this creates new collaboration opportunities that go beyond logistic restrictions such as the location and attendance of performers.

Wiggins (2006a) formalises Boden (2015)'s definition of creativity with a framework called Creative Systems Framework (CSF). CSF also includes a conceptual space, a rule set that defines the conceptual space, a rule set that defines how agents can traverse the space, an evaluation rule set that assesses the value and novelty of concepts. Wiggins (2006a) concludes that exploratory creativity at the meta-level is, in fact, transformational creativity. Hence, Wiggins (2006a) emphasises search approaches in Computational Creativity studies.

VMO, *FILTER* and *MASOM* are three systems that apply the idea of search in the conceptual space. These systems define a conceptual, multi-dimensional musical space. The dimensions of the space are audio features, i.e. sound properties. *MASOM* applies VMM, and *FILTER* implements FO for statistical sequence modelling. VMO is a model that combines VMM with FO. Following their work on VOM, Wang and Dubnov (2017) combine HMM and VMO for the MuMe task of harmony generation. Although this work applies style imitation with symbolic representation of music, Wang and Dubnov (2017) compare VMO with HMM-GMM and *K*-Means machine learning algorithms. Wang and Dubnov (2017) conclude that in the conceptual feature space, VMO models temporal relationships whereas HMM-GMM and K-Means clusters spatially.

VMO, FILTER and *MASOM* define the musical form as a traversal in this multi-dimensional space. Since we can mathematically model a traversal in a multi-dimensional space, two Metacreative opportunities arise: style combination and style transformation. Style combination is combining different attributions of styles to come up with a new style. This corresponds to combinational creativity in Boden's taxonomy of creativity. However, we do not know if the process of combining styles is linear in mathematical forms. If we combine two styles, do we explore a region that is an intersection for these two styles? A generalised version of this question is, how do we traverse the conceptual musical space by combining different attributions of two styles? Style transformation is applying a transformation function to a style to come up with another style or a new style. If we can model musical form and musical style mathematically, we can also define a function that transforms one style to another. We can also explore variety of transformation functions to research new styles.

The algorithms that we cover in this survey introduce biases and some of these biases have been pointed out in the literature. Thomas et al. (2013) compares Markov Models, FO, and *MusiCOG* ⑧ on the task of melody generation and concluded that Markov Models and FO deviated from the training corpus. This indicates that the authors of Metacreative systems may introduce biases to the creative output of their systems by choosing one algorithm over others. Further research is required to clarify what kind of transformations machine learning algorithms introduce to musical agents as well as MuMe systems. Then, we could approach these algorithms as transformation functions that we apply on musical creative tasks.

9. Conclusion

Autonomous computational systems have been applied to various musical tasks. MAS and Artificial Intelligence technologies exemplify autonomy in computational systems. Musical agents utilise artificial agent architectures and MAS to automatise musical tasks of composition, assisted composition, interpretation, improvisation, accompaniment, melody, rhythm, and harmony generation, continuation, style imitation, arrangement, curation. We surveyed 78 musical agents systems whose architectures have been presented in peer-reviewed platforms. We proposed a typology of musical agents that is framed around the terminologies of Generative Music, Computational Creativity, Artificial Intelligence, Metacreation and Musical Metacreation fields. This typology presents musical agents in nine dimensions of agent architectures, musical tasks, environment types, number of agents, number of agent roles, communication types, corpus types, input/output types and human interaction modality. Our survey has given the details of musical agents by grouping the systems according to the architecture types. We incorporated the architecture types of cognitive, reactive and hybrid architectures in MAS to classify musical agents. We further categorised musical agents using the architecture model paradigms. As a special case, we also used environment types to further group the reactive musical agents. Within each section, we grouped musical agents by the musical task that they carry out. We hope that this organisation of the survey guides the reader to have an understanding of what has been done in the interdisciplinary field of musical agents.

We mentioned in Section 2 that creative tasks lack quality measures, which highlights the difficulties of evaluating musical agents. We suggested a classification of evaluation of musical agents where the specific evaluation methodologies of systems that we survey is incorporated to the corresponding classes. We ended this section by highlighting a possibility of benchmarking tasks for musical agents. We started our final section with an overlook of architectures and algorithms that have been covered by musical agents, which indicate the opportunities of different architecture types that can be applied to musical agents. Towards our conclusion, we introduce Musical Metacreation as a field, mention the interdisciplinarity of the field and design consideration of MuMe systems. We proposed six levels of musical agents, which is a derivation of seven levels of MuMe systems proposed in the literature. Before conclusion, we remark the challenges and opportunities in musical agents, and indicate several reasons why the research of musical agents as well as MuMe matters.

The studies of MuMe field aim to guide musicians and artists to understand musical creativity and find new ways of musical creativity. We hope that this review of musical agents helps both researchers and practitioners to understand and design autonomous software making music. Almost all studies mentioned in this review are presented in the last two decades. With the increasing research on MAS and AI, we are confident that musical agents will influence and contribute to how we make music in the future.

Acknowledgements

We thank the anonymous reviewers for their insights and useful comments. We would like to thank our colleagues Ronald Boersen, Mirjana Prpa, Jianyu Fan, and Cale Plut for their suggestions and proofreading this paper. We also thank Ronald Boersen for helping us with Figure 2.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This research was funded by the Natural Sciences and Engineering Research Council of Canada Discovery programme, and Social Sciences and Humanities Research Council of Canada Insight programme.

ORCID

Kıvanç Tatar http://orcid.org/0000-0003-4133-8641

References

Al-Rifaie, A. M., & Al-Rifaie, M. M. (2015). Generative music with stochastic diffusion search. In C. Johnson, A. Carballal, & J. Correia (Eds.), *Evolutionary and biologically inspired music, sound, art and design* (pp. 1–14). Lecture Notes in Computer Science, Vol. 9027. Springer International.

Allik, A. (2014). Gene expression synthesis. In *Proceedings of the joint conference ICMC14-SMC14*.

Ámorim, A., Góes, L. F. W., da Silva, A. R., & Franĉsa, C. (2017). Creative flavor pairing: Using RDC metric to generate and assess ingredients combinations. In *Proceedings of the eight international conference on computational creativity (ICCC 2017)*.

Ando, D., & Iba, H. (2005). Real-time musical interaction between musician and multi-agent system. In *Proceedings of the 8th generative art conference*.

Arel, I., Rose, D. C., & Karnowski, T. P. (2010). Deep machine learning – a new frontier in artificial intelligence research. *IEEE Computational Intelligence Magazine*, *5*(4), 13–18.

Arges, K., Forth, J., & Wiggins, G. A. (2016). Evaluation of musical creativity and musical metacreation systems. *Computers in Entertainment (CIE) – Special Issue on Musical Metacreation, Part II*, *14*(3

Arias, J., Desainte-Catherine, M., & Dubnov, S. (2016). Automatic construction of interactive machine improvisation scenarios from audio recordings. In *The fourth international workshop on musical metacreation (MUME 2016)*.

Assayag, G., Bloch, G., Chemillier, M., Cont, A., & Dubnov, S. (2006). Omax brothers: A dynamic topology of agents for improvization learning. In *Proceedings of the 1st ACM workshop on audio and music computing multimedia* (pp. 125–132). ACM Press.

Assayag, G., & Dubnov, S. (2004). Using factor oracles for machine improvisation. *Soft Computing*, *8*(9), 604–610.

Assayag, G., Dubnov, S., & Delerue, O. (1999). Guessing the composer's mind: Applying universal prediction to musical style. In *Proceedings of the 1999 international computer music conference, ICMC 1999*, 6.

Aucouturier, J.-J. (2011). Artificial evolution of tuning systems. In *A-life for music: Music and computer models of living systems*. A-R Editions.

Aucouturier, J.-J., & Pachet, F.. (2005). Ringomatic: A real-time interactive drummer using constraint-satisfaction and drum sound descriptors. In *Proceedings of the international conference on music information retrieval* (pp. 412–419).

Baltazar, P., de la Hogue, T., & Desainte-Catherine, M.. (2014). Demo: i-score, an interactive sequencer for the intermedia arts.

Bernardo, F., Zbyszynski, M., Fiebrink, R., & Grierson, M. (2016). Interactive machine learning for end-user innovation. In *Proceedings of AAAI spring symposium*. American Association for Artificial Intelligence (AAAI).

Bevilacqua, F., Zamborlin, B., Sypniewski, A., Schnell, N., Guédy, F., & Rasamimanana, N. (2009). Continuous realtime gesture following and recognition. In *International gesture workshop* (pp. 73–84). Springer.

Beyls, P. (2007). Interaction and self-organisation in a society of musical agents. In *Proceedings of ECAL 2007 workshop on music and artificial life (MusicAL 2007)*.

Beyls, P. (2008). On-line development of man-machine relationships: Motivation-driven musical interaction. In *Proceedings of the 11th generative art conference*.

Beyls, P. (2009). Interactive composing as the expressions of autonomous machine motivations. In *Proceedings of the international computer music conference (ICMC 2009)*.

Beyls, P. (2011). Structural coupling in a society of musical agents. In *A-life for music: Music and computer models of living systems*. A-R Editions.

Beyls, P. (2012). Autonomy, influence and emergence in an audiovisual ecosystem. In *Proceedings of the generative arts conference*, Rome, Italy.

Beyls, P., Bernardes, G., & Caetano, M. (2015). EarGram actors: An interactive audiovisual system based on social behavior. *Journal of Science and Technology of the Arts*, *7*(1), 43–54.

Biles, J. (1994). GenJam: A genetic algorithm for generating jazz solos. In *Proceedings of the international computer music conference* (pp. 131–131). International Computer Music Association.

Biles, J. A. (2013). Performing with technology: Lessons learned from the GenJam project. In *Proceedings of the 2nd international workshop on musical metacreation (MUME 2013)*.

Bizzocchi, J., Eigenfeldt, A., & Thorogood, M. (2015). Generating affect: Applying valence and arousal values to unified video, music, and sound generation system. In *Proceedings of the 18th generative art conference* (Vol. 49, pp. 621–630).

Blackwell, T., Bown, O., & Young, M. (2012). Live algorithms: Towards autonomous computer improvisers. In J. McCormack, & M. d'Inverno (Eds.), *Computers and creativity* (pp. 147–174). Berlin: Springer.

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

Bloch, G., Dubnov, S., & Assayag, G. (2008). Introducing video features and spectral descriptors in the omax improvisation system. In *International computer music conference '08*.

Boden, M. A. (2009). Computer models of creativity. *AI Magazine*, *30*(3), 23.

Boden, M. A. (2015). Creativity and ALife. *Artificial Life*, *21*(3), 354–365.

Bodily, P. M., & Ventura, D. (2018). Musical metacreation: Past, present, and future. In *Proceedings of the sixth international workshop on musical metacreation* (p. 5).

Bown, O. (2011). Experiments in modular design for the creative composition of live algorithms. *Computer Music Journal*, *35*(3), 73–85.

Bown, O., Carey, B., & Eigenfeldt, A. (2015). Manifesto for a musebot ensemble: A platform for live interactive performance between multiple autonomous musical agents. In *Proceedings of the international symposium of electronic art 2015 (ISEA 2015)*.

Bown, O., & Martin, A. (2013). Backgammon: Process-based musical explorations using the agent designer. In *Proceedings of the 9th ACM conference on creativity & cognition, C&C '13* (pp. 390–391). New York, NY: ACM Press.

Bown, O., McCormack, J., & Kowaliw, T. (2011). Ecosystemic methods for creative domains: Niche construction and boundary formation. In *2011 IEEE symposium on artificial life (ALIFE)* (pp. 132–139).

Bray, L., & Bown, O. (2014). Linear and non-linear composition systems: User experience in nodal and pro tools. In *Proceedings of the Australian computer music association conference*.

Bray, L., & Bown, O. (2016). Applying core interaction design principles to computational creativity. In *Proceedings of the seventh international conference on computational creativity*.

Bray, L., Bown, O., & Carey, B. (2017). How can we deal with the design principle of visibility in highly encapsulated computationally creative systems? In *Proceedings of the eighth international conference on computational creativity*.

Bretan, M., & Weinberg, G. (2016). A survey of robotic musicianship. *Communications of the ACM*, *59*(5), 100–109.

Briot, J.-P., Hadjeres, G., & Pachet, F. (2017). Deep learning techniques for music generation – a survey. *arXiv preprint arXiv:1709.01620*.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, *2*(1), 14–23.

Brooks, R. A. (1995). Intelligence without reason. In L. Steels & R. A. Brooks (Eds.), *The artificial life route to artificial intelligence: Building embodied, situated agents* (pp. 25–81). Hillsdale, NJ: L. Erlbaum Associates.

Bryson, J. (1995). The reactive accompanist: Adaptation and behavior decomposition in a music system. In L. Steels (Ed.), *The biology and technology of intelligent autonomous agents* (pp. 365–376). Berlin: Springer.

Buchanan, B. G. (2001). Creativity at the metalevel AAAI-2000 presidential address. *AI Magazine*, *22*(3), 16.

Camurri, A., Catorcini, A., Innocenti, C., & Massari, A. (1995). Music and multimedia knowledge representation and reasoning: The HARP system. *Computer Music Journal*, *19*(2), 34–1.

Casella, P., & Paiva, A. (2001). Magenta: An architecture for real time automatic composition of background music. In *Intelligent virtual agents* (pp. 224–232). Springer.

Cassell, J., Sullivan, J., Churchill, E., & Prevost, S. (2000). *Embodied conversational agents*. London: MIT Press. Google-Books-ID: tHiKZGh9t7sC.

Collins, N. (2005). Drumtrack: Beat induction from an acoustic drum kit with synchronised scheduling. In *Proceedings of international computer music conference (ICMC)*.

Collins, N. (2006). BBCut2: Integrating beat tracking and on-the-fly event analysis. *Journal of New Music Research*, *35*(1), 63–70.

Collins, N. (2008). Reinforcement learning for live musical agents. In *Proceedings of the international computer music conference (ICMC), Belfast*.

Collins, N. (2011). *LL: Listening and learning in an interactive improvisation system*. Technical report, University of Sussex.

Collins, N. (2017). Towards machine musicians who have listened to more music than us: Audio database-led algorithmic criticism for automatic composition and live concert systems. *Computers in Entertainment*, *14*(3), 1–14.

Colton, S. (2008). Creativity versus the perception of creativity in computational systems. In *AAAI spring symposium: Creative intelligent systems* (Vol. 8).

Colton, S., & Wiggins, G. A. (2012). Computational creativity: The final frontier. *Frontiers in Artificial Intelligence and Applications*, *242*, 21–26.

Conklin, D. (2013). Multiple viewpoint systems for music classsification. *Journal of New Music Research*, *42*(1), 19–26.

Cont, A., Dubnov, S., & Assayag, G. (2007). Anticipatory model of musical style imitation using collaborative and competitive reinforcement learning. In *Anticipatory behavior in adaptive learning systems* (pp. 285–306). Springer.

Csikszentmihalyi, M. (2008). *Flow: The psychology of optimal experience*. (1st ed.). New York, NY: Harper Perennial Modern Classics.

Dahlstedt, P., & Nordahl, M. G. (2001). Living melodies: Coevolution of sonic communication. *Leonardo*, *34*(3), 243–248.

Delgado, M., Fajardo, W., & Molina-Solana, M. (2009). Inmamusys: Intelligent multiagent music system. *Expert Systems with Applications*, *36*(3), 4574–4580.

Déguernel, K., Vincent, E., & Assayag, G. (2018). Probabilistic factor oracles for multidimensional machine improvisation. *Computer Music Journal*, *42*(2), 52–66.

Donze, A., Valle, R., Akkaya, I., Libkind, S., Seshia, S. A., & Wessel, D. (2014). Machine improvisation with formal specifications. In *Proceedings of the joint conference ICMC14-SMC14* (p. 8).

Dubnov, S., & Assayag, G. (2005). Improvisation planning and jam session design using concepts of sequence variation and flow experience. In *Proceedings of sound and music computing* (p. 7).

Dubnov, S., Assayag, G., & Cont, A. (2007). Audio oracle: A new algorithm for fast learning of audio structures. In *Proceedings of international computer music conference*.

Dubnov, S., Assayag, G., & Cont, A. (2011). Audio oracle analysis of musical information rate. In *Proceedings of the fifth IEEE international conference on semantic computing (ICSC)* (pp. 567–571).

Dubnov, S., Assayag, G., & El-Yaniv, R. (1998). Universal classification applied to musical sequences. In *Proceedings of the 1998 international computer music conference, ICMC*.

Dubnov, S., McAdams, S., & Reynolds, R. (2006). Structural and affective aspects of music from statistical audio signal analysis. *Journal of the American Society for Information Science and Technology*, *57*(11), 1526–1536.

Eerola, T., & Vuoskoski, J. K. (2013). A review of music and emotion studies: Approaches, emotion models, and stimuli. *Music Perception: An Interdisciplinary Journal*, *30*(3), 307–340.

Eigenfeldt, A. (2008). Emergent rhythms through multi-agency in Max/MSP. In R. Kronland-Martinet, S. Ystad, & K. Jensen (Eds.), *Computer music modeling and retrieval. Sense of sounds* (pp. 368–379). Lecture Notes in Computer Science, Vol. 4969. Berlin: Springer.

Eigenfeldt, A. (2009). The evolution of evolutionary software: Intelligent rhythm generation in kinetic engine. In M. Giacobini, A. Brabazon, S. Cagnoni, G. A. D. Caro, A. Ekárt, A. I. Esparcia-Alcázar, M. Farooq, A. Fink, & P. Machado (Eds.), *Applications of evolutionary computing* (pp. 498–507). Lecture Notes in Computer Science, Vol. 5484. Berlin: Springer.

Eigenfeldt, A. (2010). Coming together: Composition by negotiation. In *Proceedings of the 18th ACM international conference on multimedia* (pp. 1433–1436). ACM.

Eigenfeldt, A. (2011). Multi-agent modeling of complex rhythmic interactions in realtime performance. In *A-life for music: Music and computer models of living systems*. A-R Editions.

Eigenfeldt, A. (2014). Generating structure-towards large-scale formal generation. In *Proceedings of the artificial intelligence and interactive digital entertainment conference*.

Eigenfeldt, A. (2016a). Exploring moment-form in generative music. In *Proceedings of 13th sound and music conference*.

Eigenfeldt, A. (2016b). Musebots at one year: A review. In *Proceedings of the 4th international workshop on musical metacreation (MUME 2016)*.

Eigenfeldt, A., Bown, O., & Carey, B. (2015). Collaborative composition with creative systems: Reflections on the first musebot ensemble. In *Proceedings of the sixth international conference on computational creativity*, June (p. 134).

Eigenfeldt, A., Bown, O., Pasquier, P., & Martin, A. (2013). Towards a taxonomy of musical metacreation: Reflections on the first musical metacreation weekend. In *Proceedings of the 2nd international workshop on musical metacreation (MUME 2013)*.

Eigenfeldt, A., & Pasquier, P. (2009). A realtime generative music system using autonomous melody, harmony, and rhythm agents. In *Proceedings of the 12th generative art conference*.

Eigenfeldt, A., & Pasquier, P. (2011a). Negotiated content: Generative soundscape composition by autonomous musical agents in coming together: Freesound. In *Proceedings of the second international conference on computational creativity*, Mexico City (pp. 27–32).

Eigenfeldt, A., & Pasquier, P. (2011b). A sonic eco-system of self-organising musical agents. In *9th European event on evolutionary and biologically inspired music, sound, art and design (EvoMusArt 2011)* (Vol. 6625, pp. 283–292). Torino: Springer Verlag.

Eigenfeldt, A., & Pasquier, P. (2012). Creative agents, curatorial agents, and human-agent interaction in coming together. In *Proceedings of sound and music computing* (pp. 181–186).

Einbond, A., Borghesi, R., Schwarz, D., & Schnell, N. (2016). Introducing CatOracle: Corpus-based concatenative improvisation with the Audio Oracle algorithm. In *Proceedings of the international computer music conference* (pp. 140–146).

Emirbayer, M., & Mische, A. (1998). What is agency. *American Journal of Sociology*, *103*(4), 962–1023.

Fan, J., Tatar, K., Thorogood, M., & Pasquier, P. (2017). Ranking-based emotion recognition for experimental music. In *Proceedings of the international symposium on music information retrieval (ISMIR) 2017*.

Ferber, J., Gutknecht, O., & Michel, F. (2003). From agents to organizations: An organizational view of multi-agent systems. In *International workshop on agent-oriented software engineering* (pp. 214–230). Springer.

Fowler, C. B. (1967). The museum of music: A history of mechanical instruments. *Music Educators Journal*, *54*(2), 45.

Franois, A. R. J., Chew, E., & Thurmond, D. (2011). Performer-centered visual feedback for human-machine improvisation. *Computers in Entertainment*, *9*(3), 1–13.

Franois, A. R., Schankler, I., & Chew, E. (2013). Mimi4x: An interactive audio-visual installation for high-level structural improvisation. *International Journal of Arts and Technology*, *6*(2), 138–151.

Frayling, C. (1994). *Research in art and design (Royal college of art research papers, Vol 1, No 1, 1993/4)*. Royal College of Art Research Papers 1(1).

Galanter, P. (2003). What is generative art complexity theory as a context for art theory. In *Proceedings of the 6th generative art conference*.

Garcia, R. (2001). *Automatic generation of sound synthesis techniques* (Ph.D. Dissertation). MIT.

George, D. (2008). *How the brain might work: A hierarchical and temporal model for learning and recognition* (Ph.D. Dissertation). Stanford University.

Gifford, T. (2013). Appropriate and complementary rhythmic improvisation in an interactive music system. In S. Holland, K. Wilkie, P. Mulholland, & A. Seago (Eds.), *Music and human-computer interaction. Springer Series on Cultural Computing. London: Springer London*.

Gifford, T. M., & Brown, A. R. (2010). Anticipatory timing in algorithmic rhythm generation. In *Proceedings of the Australasian computer music conference 2010* (pp. 21–28). Australasian Computer Music Association (ACMA).

Gillies, M., Lee, B., d'Alessandro, N., Tilmanne, J., Kulesza, T., Caramiaux, B., … Amershi, S. (2016). *Human-centred machine learning* (pp. 3558–3565). New York, NY: ACM Press.

Gimenes, M., & Miranda, E. R. (2011). An ontomemetic approach to musical intelligence. In *A-life for music: Music and computer models of living systems*. A-R Editions.

Gimenes, M., Miranda, E. R., & Johnson, C. (2005). Towards an intelligent rhythmic generator based on given examples: A memetic approach. In *Digital music research network summer conference* (pp. 41–46).

Gimenes, M., Miranda, E. R., & Johnson, C. (2007). Musicianship for robots with style. In *Proceedings of the 7th international conference on new interfaces for musical expression* (pp. 197–202). ACM.

Gold, R., & Maeda, J. (2007). *The plenitude: Creativity, innovation, and making stuff*. Cambridge: The MIT Press.

Gomila, A., & Müller, V. C. (2012). Challenges for artificial cognitive systems. *Journal of Cognitive Science*, *13*(4), 453–469.

Harding, S., Leitner, J., & Schmidhuber, J. (2013). Cartesian genetic programming for image processing. In R. Riolo, E. Vladislavleva, M. D. Ritchie, & J. H. Moore (Eds.), *Genetic programming theory and practice X, genetic and evolutionary computation* (pp. 31–44). New York: Springer. doi:10.1007/978-1-4614-6846-2_3.

Hartholt, A., Traum, D., Marsella, S. C., Shapiro, A., Stratou, G., Leuski, A., … Gratch, J. (2013). All together now: Introducing the virtual human toolkit. In *13th international conference on intelligent virtual agents*.

Hawryshkewich, A., Pasquier, P., & Eigenfeldt, A. (2010). Beatback: A real-time interactive percussion system for rhythmic practise and exploration. *Proceedings of the tenth international conference on new interfaces for musical expression* (pp. 100–105).

Herremans, D., Chuan, C.-H., & Chew, E. (2017). A functional taxonomy of music generation systems. *ACM Computing Surveys*, *50*(5), 1–30.

Heylighen, F. (2016). Stigmergy as a universal coordination mechanism I: Definition and components. *Cognitive Systems Research*, *38*, 4–13.

Hsu, W. (2010). Strategies for managing timbre and interaction in automatic improvisation systems. *Leonardo Music Journal*, *20*(1), 33–39.

Huron, D. B. (2014). *Sweet anticipation: Music and the psychology of expectation*. Cambridge: MIT Press.

Jordanous, A. (2012). A standardised procedure for evaluating creative systems: Computational creativity evaluation based on what it is to be creative. *Cognitive Computation*, *4*(3), 246–279.

Kimball, J. P. (1975). *Syntax and semantics*. Cambridge, MA: Academic Press.

Kirke, A., & Miranda, E. (2011). A biophysically constrained multi-agent systems approach to algorithmic composition with expressive performance. In *A-life for music: Music and computer models of living systems*. A-R Editions.

Kirke, A., & Miranda, E. (2015). A multi-agent emotional society whose melodies represent its emergent social hierarchy and are generated by agent communications. *Journal of Artificial Societies and Social Simulation*, *18*(2), 16.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*(1), 59–69.

Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, *21*(1-3), 1–6.

Lartillot, O., Cereghetti, D., Eliard, K., & Grandjean, D. (2013). A simple, high-yield method for assessing structural novelity. In G. Luck & O. Brabant (Eds.), *Proceedings of the 3rd international conference on music & emotion (ICME3), Jyväskylä, Finland, 11th–15th June 2013*. ISBN 978-951-39-5250-1. University of Jyväskylä, Department of Music.

Lefebvre, A., & Lecroq, T. (2002). A heuristic for computing repeats with a factor oracle: Application to biological sequences. *International Journal of Computer Mathematics*, *79*(12), 1303–1315.

Levisohn, A., & Pasquier, P. (2008). BeatBender: Subsumption architecture for autonomous rhythm generation. In *Proceedings of the ACM international conference on advances in computer entertainment technologies (ACE 2008)* (pp. 51–58).

Lévy, B., Bloch, G., & Assayag, G. (2012). OMaxist dialectics. In *Proceedings of the international conference on new interfaces for musical expression (NIME)* (pp. 137–140).

Lewis, G. E. (2000). Too many notes: Computers, complexity and culture in voyager. *Leonardo Music Journal*, *10*, 33–39.

Linson, A., Dobbyn, C., Lewis, G. E., & Laney, R. (2015). A subsumption agent for collaborative free improvisation. *Computer Music Journal*, *39*(4), 96–115.

Lynch, M. F. (2014). Motivation, microdrives and microgoals in mockingbird. In *Proceedings of 3rd international workshop on musical metacreation (MUME 2014)*.

Macret, M., & Pasquier, P. (2014). Automatic design of sound synthesizers as pure data patches using coevolutionary mixed-typed cartesian genetic programming. In *Proceedings of the 2014 conference on genetic and evolutionary computation, GECCO '14* (pp. 309–316). New York, NY: ACM.

Martin, A., & Bown, O. (2013). The agent designer toolkit. In *Proceedings of the 9th ACM conference on creativity & cognition, C&C '13* (pp. 386–387). New York, NY: ACM.

Martin, A., Jin, C. T., & Bown, O. (2011). A toolkit for designing interactive musical agents. In *Proceedings of the 23rd Australian computer-human interaction conference, OzCHI '11* (pp. 194–197). New York, NY: ACM.

Martin, A., Jin, C. T., & Bown, O. (2012). Implementation of a real-time musical decision-maker. In *Proceedings of the Australasian computer music conference*.

Martin, A., Jin, C. T., Carey, B., & Bown, O. (2012). Creative experiments using a system for learning high-level performance structure in ableton live. In *Proceedings of the sound and music computing conference*.

Martin, A., Jin, C. T., van Schaik, A., & Martens, W. L. (2010). Partially observable Markov decision processes for interactive music systems. In *Proceedings of the international computer music conference*.

Martins, J. M., & Miranda, E. R. (2006). A connectionist architecture for the evolution of rhythms. In *Applications of evolutionary computing* (pp. 696–706). Springer.

Martins, J. M., & Miranda, E. R. (2007). Emergent rhythmic phrases in an A-Life environment. In *Proceedings of ECAL 2007 workshop on music and artificial life (MusicAL 2007)* (pp. 10–14).

Martins, J. M., & Miranda, E. R. (2008). Breeding rhythms with artificial life. In *Proceedings of the sound and music conference*. Citeseer.

Maxwell, J.B., Eigenfeldt, A., Pasquier, P., Gonzalez Thomas, N. (2012). MusiCOG: A cognitive architecture for music learning and generation. In *Proceedings of the sound and music computing conference* (p. 9).

Maxwell, J. B., Pasquier, P., & Eigenfeldt, E. (2009). Hierarchical sequential memory for music: A cognitive model. In *Proceedings of the 10th international conference for music information retrieval*.

McCormack, J., & Bown, O. (2009). Life's what you make: Niche construction and evolutionary art. In *Workshops on applications of evolutionary computation* (pp. 528–537). Springer.

McCormack, J., McIlwain, P., Lane, A., & Dorin, A. (2007). Generative composition with Nodal. In *Workshop on music and artificial life (part of ECAL 2007), Lisbon, Portugal*.

Miller, J. F.ed (2011). *Cartesian genetic programming*. Natural Computing Series. Berlin: Springer.

Minsky, M. (1986). *The society of mind*. New York, NY: Simon and Schuster.

Miranda, E. R., & Biles, A. (Eds.). (2007). *Evolutionary computer music*. London: Springer.OCLC: ocm80332658.

Miranda, E. R., Kirke, A., & Zhang, Q. (2010). Artificial evolution of expressive performance of music: An imitativemulti-agent systems approach. *Computer Music Journal*, *34*(1), 80–96.

Mitchell, T. M. (1997). *Machine learning*. New York, NY: McGraw-Hill Education.

Moreira, J., Roy, P., & Pachet, F. (2013). Virtualband: Interacting with stylistically consistent agents. In *Proceedings of the 14th international society for music information retrieval conference* (pp. 341–346).

Müller, M. (2015). *Fundamentals of music processing*. Cham: Springer International.

Murray-Rust, D. (2008). *Musical acts and musical agents: Theory, implementation and practice* (Ph.D. dissertation).

Murray-Rust, D., & Smaill, A. (2005). Musical acts and musical agents. *Proceedings of the 5th open workshop of MUSICNETWORK: Integration of music in multimedia applications (to Appear)* 10.

Murray-Rust, D., & Smaill, A. (2011). Towards a model of musical interaction and communication. *Artificial Intelligence*, *175*(9–10), 1697–1721.

Murray-Rust, D., Smaill, A., & Edwards, M. (2006). MAMA: An architecture for interactive musical agents. *Frontiers in Artificial Intelligence and Applications*, *141*, 36.

Murray-Rust, D., Smaill, A., & Maya, M. (2005). VirtuaLatin – towards a musical multi-agent system. In *Sixth international conference on computational intelligence and multimedia applications, 2005* (pp. 17–22).

Navarro, M., Corchado, J. M., & Demazeau, Y. (2014). A musical composition application based on a multiagent system to assist novel composers. *International conference on computational creativity*.

Navarro, M., Corchado, J. M., & Demazeau, Y. (2016). MUSIC-MAS: Modeling a harmonic composition system with virtual organizations to assist novice composers. *Expert Systems with Applications*, *57*, 345–355.

Nika, J., Bouche, D., Bresson, J., Chemillier, M., & Assayag, G. (2015). Guided improvisation as dynamic calls to an offline model. In *Sound and music computing (SMC)*.

Nika, J., & Chemillier, M. (2012). Improtek: integrating harmonic controls into improvisation in the filiation of OMax. In *International computer music conference (ICMC)* (pp. 180–187).

Nika, J., Chemillier, M., & Assayag, G. (2017). ImproteK: Introducing scenarios into human-computer music improvisation. *Computers in Entertainment*, *14*(2), 1–27.

Nika, J., Echeveste, J., Chemillier, M., & Giavitto, J.-L. (2014). Planning human-computer improvisation. In *International computer music conference* (p. 330).

Nort, D. V., Oliveros, P., & Braasch, J. (2013). Electro/acoustic improvisation and deeply listening machines. *Journal of New Music Research*, *42*(4), 303–324.

Pachet, F. (2000). Rhythms as emerging structures. In *Proceedings of 2000 international computer music conference, Berlin, ICMA*.

Pachet, F. (2003). The continuator: Musical interaction with style. *Journal of New Music Research*, *32*(3), 333–341.

Pachet, F. (2004). Beyond the cybernetic jam fantasy: The continuator. *Computer Graphics and Applications, IEEE*, *24*(1), 31–35.

Paiva, A., Andersson, G., Höök, K., Mourão, D., Costa, M., & Martinho, C. (2002). Sentoy in fantasya: Designing an affective sympathetic interface to a computer game. *Personal and Ubiquitous Computing*, *6*(5-6), 378–389.

Pasquier, P., Eigenfeldt, A., Bown, O., & Dubnov, S. (2017). An introduction to musical mestacreation. *Computers in Entertainment*, *14*(2), 1–14.

Pease, A., & Colton, S. (2011). On impact and evaluation in computational creativity: A discussion of the Turing test and an alternative proposal. In *Proceedings of the AISB symposium on AI and philosophy*.

Peter, M. (2009). Milieus of creativity: The role of places, environments, and spatial. In P. Meusburger, J. Funke, & E. Wunder (Eds.), *Knowledge and space; Vol. 2. Milieus of creativity: An interdisciplinary approach to spatiality of creativity* (pp. 97–153). Dordrecht: Springer.

Plans, D., & Morelli, D. (2011). Using coevolution in music improvisation. In *A-life for music: Music and computer models of living systems*. A-R Editions.

Plomp, R., & Levelt, W. J. M. (1965). Tonal consonance and critical bandwidth. *The Journal of the Acoustical Society of America*, *38*(4), 548–560.

Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. Wiley series in probability and mathematical statistics. Applied Probability and Statistics Section. New York: John Wiley & Sons.

Rigau, J., Feixas, M., & Sbert, M. (2008). Informational aesthetics measures. *IEEE Computer Graphics and Applications*, *28*(2), 24–34.

Ritchie, G. (2014). *Evaluating quality in creative systems*.

Roads, C. (2015). *Composing electronic music: A new aesthetic*. Oxford: Oxford University Press.

Rowe, R. (1992). Machine listening and composing with cypher. *Computer Music Journal*, *16*(1), 43.

Russell, J. A. (1980). A circumplex model of affect. *Journal of Personality and Social Psychology*, *39*(6), 1161–1178.

Russell, S. J. S. J., & Norvig, P. (2010). *Artificial intelligence: A modern approach*. (3rd ed.). Prentice Hall series in artificial intelligence. Upper Saddle River, NJ: Prentice Hall.

Sampaio, P. A., Ramalho, G., & Tedesco, P. (2008). CinBalada: A multiagent rhythm factory. *Journal of the Brazilian Computer Society*, *14*(3), 31–49.

Simon, H. A. (1960). *The new science of management decision*. The Ford Distinguished Lectures, Vol. xii. New York, NY: Harper & Brothers. doi:10.1037/13978-000.

Sivanandam, S. N., & Deepa, S. N. (2007). *Introduction to genetic algorithms*. Berlin: Springer.

Smalley, D. (1997). Spectromorphology: Explaining sound-shapes. *Organised Sound*, *2*(02), 107–126.

Smith, B. D., & Deal, W. S. (2014). ML.* Machine learning library as a musical partner in the computer-acoustic composition flight. In *the Proceedings of the joint conference ICMC14-SMC14* (Vol. 2014).

Smith, B. D., & Garnett, G. E. (2012). Reinforcement learning and the creative, automated music improviser. In P. Machado, J. Romero, & A. Carballal (Eds.), *Evolutionary and biologically inspired music, sound, art and design* (pp. 223–234). Lecture Notes in Computer Science, Vol. 7247. Berlin: Springer.

Sumpter, D. J., & Beekman, M. (2003). From nonlinearity to optimality: Pheromone trail foraging by ants. *Animal Behaviour*, *66*(2), 273–280.

Surges, G., & Dubnov, S. (2013). Feature selection and composition using PyOracle. In *Ninth artificial intelligence and interactive digital entertainment conference*.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Adaptive Computation and Machine Learning. Cambridge, MA: A Bradford Book.

Takala, T., Hahn, J., Gritz, L., Geigel, J., & Lee, J. (1993). Using physically based models and genetic algorithms for functional composition of sound signals, synchronized to animated motion. In *Proceedings of the international computer music conference* (pp. 180–185).

Tatar, K., Macret, M., & Pasquier, P. (2016). Automatic synthesizer preset generation with presetGen. *Journal of New Music Research*, *45*(2), 124–144.

Tatar, K., & Pasquier, P. (2017). MASOM: A musical agent architecture based on self organizing maps, affective computing, and variable Markov models. In *Proceedings of the 5th international workshop on musical metacreation (MUME 2017)*.

Tatar, K., Pasquier, P., & Siu, R. (2018). *REVIVE: An audio-visual performance with musical and visual AI agents* (pp. 1–6). New York, NY: ACM Press.

Thom, B. (2000a). BoB: An interactive improvisational music companion. In *Proceedings of the fourth international conference on autonomous agents, AGENTS '00* (pp. 309–316). New York, NY: ACM.

Thom, B. (2000b). Unsupervised learning and interactive jazz/blues improvisation. In *Proceedings of the seventeenth national conference on artificial intelligence and twelfth conference on innovative applications of artificial intelligence* (pp. 652–657).

Thom, B. (2003). Interactive improvisational music companionship: A user-modeling approach. *User Modeling and User-Adapted Interaction*, *13*(1-2), 133–177.

Thomas, N. G., Pasquier, P., Eigenfeldt, A., & Maxwell, J. B. (2013). A methodology for the comparison of melodic generation models using meta-melo. In *Proceedings of the 14th international society for music information retrieval conference* (pp. 561–566).

Thórisson, K., & Helgasson, H. (2012). Cognitive architectures and autonomy: A comparative review. *Journal of Artificial General Intelligence*, *3*(2), 1–30.

Todd, P. M., & Werner, G. M. (1999). Frankensteinian methods for evolutionary music. In *Musical networks: Parallel distributed perception and performance* (pp. 313–340). Cambridge, MA: MIT Press/Bradford Books.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, *59*(236), 433–460.

Ueda, L. K., & Kon, F. (2003). Andante: A mobile musical agents infrastructure. In *Proceedings of the 9th Brazilian symposium on computer music* (pp. 87–94).

Valle, R., Donzé, A., Fremont, D. J., Akkaya, I., Seshia, S. A., Freed, A., & Wessel, D. (2017). Specification mining for machine improvisation with formal specifications. *Computers in Entertainment*, *14*(3), 1–20.

Varese, E., & Wen-chung, C. (1966). The liberation of sound. *Perspectives of New Music*, *5*(1), 11–19.

Vicari, R. M., Nakayama, L., Wulfhorst, R. D., Costalonga, L. L., & Miletto, E. M. (2005). The musical interactions within community agents. *Agent-based simulation conference*.

Vold, J. N. (1986). *A study of musical problem solving behavior in kindergarten children and a comparison with other aspects of creative behavior* (Ph.D. dissertation). University of Alabama.

Wang, C. I., & Dubnov, S. (2014). Guided music synthesis with variable Markov oracle. In *The 3rd international workshop on musical metacreation, 10th artificial intelligence and interactive digital entertainment conference*.

Wang, C. I., & Dubnov, S. (2017). Context-aware hidden Markov models of jazz music with variable Markov oracle. In *Proceedings of the 5th international workshop on musical metacreation (MUME 2017)*.

Webster, P. R. (1987). Conceptual bases for creative thinking in music. In *Music and child development* (pp. 158–174). New York, NY: Springer. doi:10.1007/978-1-4613-8698-8_8

Wehn, K. (1998). Using ideas from natural selection to evolve synthesized sounds. In *Proceedings of the digital audio effects DAFX98 workshop* (pp. 159–167).

Weiss, G. (2013). *Multiagent systems intelligent robotics and autonomous agents* (2nd ed.). Cambridge: The MIT Press.

Whalley, I. (2004). PIWeCS: Enhancing human/machine agency in an interactive composition system. *Organised Sound*, *9*(2), 167–174.

Whitelaw, M. (2004). *Metacreation: Art and artificial life*. Cambridge: MIT Press.

Wiggins, G. A. (2006a). A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems*, *19*(7), 449–458.

Wiggins, G. A. (2006b). Searching for computational creativity. *New Generation Computing*, *24*(3), 209–222.

Wooldridge, M. (2009). *An introduction to multiagent systems*. Hoboken, NJ: John Wiley & Sons.

Wulfhorst, R. D., Nakayama, L., & Vicari, R. M. (2003). A multiagent approach for musical interactive systems. In *Proceedings of the second international joint conference on autonomous agents and multiagent systems, AAMAS '03* (pp. 584–591). New York, NY: ACM.

Yee-King, M., & d'Inverno, M. (2016). Experience driven design of creative systems. In *Proceedings* of the 7th computational creativity conference (ICCC 2016). Universite Pierre et Marie Curie.

Yee-King, M. J. (2007). An automated music improviser using a genetic algorithm driven synthesis engine. In M. Giacobini (Ed.), *Applications of evolutionary computing*. Lecture Notes in Computer Science, Vol. 4448. Berlin: Springer.

Young, M. (2007). NN music: Improvising with a 'living' computer. In *International symposium on computer music modeling and retrieval* (pp. 337–350). Springer.