Musical Acts and Musical Agents: theory, implementation and practice

David Murray-Rust

Doctor of Philosophy

Centre for Intelligent Systems and their Applications

School of Informatics

University of Edinburgh

2008

Abstract

Musical Agents are an emerging technology, designed to provide a range of new musical opportunities to human musicians and composers. Current systems in this area lack certain features which are necessary for a high quality musician; in particular, they lack the ability to structure their output in terms of a communicative dialogue, and reason about the responses of their partners.

In order to address these issues, this thesis develops Musical Act Theory (MAT). This is a novel theory, which models musical interactions between agents, allowing a dialogue oriented analysis of music, and an exploration of intention and communication in the context of musical performance.

The work here can be separated into four main contributions: a specification for a Musical Middleware system, which can be implemented computationally, and allows distributed agents to collaborate on music in real-time; a computational model of musical interaction, which allows musical agents to analyse the playing of others as part of a communicative process, and formalises the workings of the Musical Middleware system; MAMA, a musical agent system which embodies this theory, and which can function in a variety of Musical Middleware applications; a pilot experiment which explores the use of MAMA and the utility of MAT under controlled conditions.

It is found that the Musical Middleware architecture is computationally implementable, and allows for a system which can respond to both direct musical communication and extramusical inputs, including the use of a custom-built tangible interface. MAT is found to capture certain aspects of music which are of interest — an intuitive notion of performative actions in music, and an existing model of musical interaction. Finally, the fact that a number of different levels — theory, architecture and implementation — are tied together gives a coherent model which can be applied to many computational musical situations.

Acknowledgements

First and foremost, I'd like to thank my supervisors, Alan Smaill and Michael Edwards for their continuing inspiration and guidance. Special thanks go to Alan for helping me turn this thesis into a coherent, presentable document. I would also like to thank the School of Informatics for assisting with my funding.

Several people have helped me make this thesis what it is, so thanks go to Paul, João, Ollie, Geraint Wiggins, Manuel, Sebastian, Michelle, Rónán, Anne, Raymond Monelle, the Dream group and the Edinburgh for ideas and critiques; Peter Nelson, Mark Steedman and Alan Bundy for guidance and support; Nicholas Ashton and the pianists at Napier, and the Edinburgh University pianists for their participation in my experiments, and Kinnell for helping me set it up; Martin Parker for helping with the conception of the AgentBox, and the Centre for Intelligent Systems and their Applications for funding it; Paul, Owen, Jules, Sebastian and Anne for reminding about the fun side of music and Alison, Sebastian and Suzy for proofreading. I'd also like to thank the creators of all the Free software which has allowed me to create this thesis.

Finally, I'd like to thank my parents, for support (of all kinds) and encouragement, and Suzy for support and unwavering psychological rigour.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*David Murray-Rust*)

Table of Contents

1 Introduction **1**

1.1 Structure of the Thesis 5

1.2 Glossary 8

2 Background **9**

2.1 Introduction 9

2.2 Musical activities and their computational, cognitive and psychological friends 10

2.2.1 Listening and Analysis 10

2.2.2 Performing 11

2.2.3 Composing 14

2.2.4 Improvising 14

2.3 Linguistic Theories - Pragmatics and Speech Act Theory 19

2.3.1 Grice’s Maxims 20

2.3.2 Speech Act Theory 21

2.3.3 Relations between music and language 23

2.4 Multiagent systems 24

2.5 Tools and frameworks for classifying and evaluating computer musical systems 25

2.5.1 Expressive Completeness and Structural Generality 25

2.5.2 Interactive Systems - Scores, Responses and Players 26

2.5.3 Net Music Approaches 27

2.5.4 Methodological Approaches and Motivation 28

2.5.5 Types of Improvisation 30

2.5.6 Creativity 31

2.5.7 Conclusions 31

3 Related Work **33**

3.1 Models of music, improvisation, interaction and creativity 33

3.1.1 A cognitive model of improvisation (Pressing) 33

3.1.2 The Generative Theory of Tonal Music 36

3.1.3 Live Algorithms 41

3.1.4 A Framework for the Analysis of Performer Interactions . . . 42

3.1.5 Other models of musical improvisation 44

3.1.6 Discussion 44

3.2 A Selection of Computer Music Systems 45

3.2.1 Constraint Satisfaction 45

3.2.2 Generative, Evolutionary and Social Systems 45

3.2.3 Net Music 46

3.2.4 Musical Companions 47

3.2.5 Intention-based systems 47

3.2.6 Musical Agent Systems 47

3.3 Three Related Works 48

3.3.1 EMI 48

3.3.2 Continuator 49

3.3.3 Andante 51

3.3.4 Positioning of the current work 53

3.4 Conclusion 53

4 Architecture and Design **55**

4.1 Musical Middleware 55

4.1.1 Motivation 59

4.1.2 Capabilities 60

4.2 Agent System Model 61

4.2.1 Environment 62

4.2.2 Realiser 62

4.2.3 System organisation 62

4.2.4 Temporal aspects 65

4.2.5 Specification 71

4.3 Musical Agent Architecture 71

4.3.1 Inputs and Outputs 73

4.3.2 Context 73

4.3.3 Deliberation 73
4.3.4 Communicative Acts 74
4.3.5 Analysis 75
4.3.6 Generation 75
4.3.7 Specification 76
4.4 Music Representation 76
4.4.1 Design Criteria 78
4.4.2 Musical Structures 80
4.5 Conclusions 82

5 Musical Acts **83**
5.1 An Intuitive Notion of Musical Acts 83
5.1.1 Characterisation of Musical Acts 84
5.1.2 An Example: Little Blue Frog 85
5.1.3 Performative Acts 87
5.2 Discussion and Future Work 88
5.3 Conclusions 89

6 A Theory of Musical Interaction **91**
6.1 Introduction 91
6.1.1 Different types of actions 91
6.2 Musical Agent System Description 93
6.2.1 State Description Overview 93
6.2.2 The Mechanics of Playing Music 96
6.3 Describing Music 98
6.3.1 Facets and Values 99
6.3.2 Relations between values 100
6.3.3 Analysis Procedures 106
6.4 Understanding other musicians 110
6.4.1 Extracting values 110
6.4.2 Temporality and Current Values 111
6.4.3 A state indexed description of music 112
6.4.4 Musical Context — the Musical Now 112
6.4.5 Common Knowledge and Musical Common Ground 113
6.5 Actions within musical context 116
6.5.1 Actions and relations 117

6.5.2 Action signatures for two agents 117
6.5.3 The universe of action signatures 119
6.5.4 A worked example 120
6.5.5 Choices 120
6.5.6 Extension to multiple agents 123
6.5.7 Reasonableness of $A_{context}$ and A_{state} 124
6.5.8 Relations to other kinds of actions 125
6.6 Relations between Musical Acts and Musical Action Signatures . . . 126
6.7 Discussion and further work 128
6.7.1 Relation to previous work 130
6.8 Conclusions 131

7 Implementation **133**
7.1 Introduction 133
7.1.1 Technologies 134
7.2 Realising Music 135
7.2.1 Musical Time, Physical Time and Beat Tracking 136
7.2.2 Converting MRA to MIDI 137
7.2.3 Turning music into sound in time 138
7.2.4 Timings for the agent system 138
7.3 Representing Music 139
7.3.1 Language Syntax 140
7.3.2 Music in Memory 143
7.3.3 Music on the Network 144
7.4 Implementing Agents 145
7.4.1 Concurrency in JADE 145
7.4.2 Main Musical Loop 146
7.4.3 Analysing and Creating Music in Agents 146
7.4.4 Bestiary of agents 148
7.5 Visualisations and Interfaces 149
7.5.1 Interfaces 149
7.5.2 MRA 149
7.5.3 Agent Space 151
7.5.4 Agent Interfaces 153
7.6 Discussion 153

7.6.1 Performance and Timing 153
7.7 Future Work 156
7.8 Conclusion 157

8 Analysis and Generation in MAMA 159
8.1 Introduction 159
8.2 Analysis 160
8.2.1 Using the Score 162
8.2.2 Features 163
8.2.3 Creating Analysers and Dependencies 164
8.3 Generation 164
8.3.1 RenderPlans and their realisation 166
8.4 The Mirroring Reasoner 167
8.4.1 Value Reasoner 167
8.5 Discussion 167
8.6 Conclusion 168

9 Musical deliberation 171
9.1 Introduction 171
9.1.1 Design Criteria 171
9.2 Deliberation formalisation 172
9.2.1 State formulation 173
9.2.2 Role of the deliberator 175
9.3 Deliberator System Design 175
9.3.1 Symbolic Values 175
9.3.2 Act Extraction 180
9.3.3 Deliberation 180
9.3.4 Instantiation 181
9.3.5 Filling the RenderPlan 185
9.4 Training the reasoner 185
9.4.1 Data Capture Setup 185
9.4.2 Analysing and Visualising the data 186
9.5 Feedback from experimental trials 186
9.6 Discussion 189
9.7 Conclusion 191

10 Case Studies - uses of the MAMA system 193

10.1 In C . . . 194

10.1.1 The piece: In C . . . 194

10.1.2 Encoding the Score . . . 195

10.1.3 Group Following . . . 196

10.1.4 Interface: AgentSpace . . . 197

10.1.5 Example Performance . . . 198

10.1.6 Discussion and Future Plans . . . 199

10.2 Canto Ostinato . . . 200

10.2.1 Encoding the score . . . 201

10.2.2 Using features . . . 202

10.2.3 Discussion . . . 202

10.3 AgentBox . . . 203

10.3.1 Computer Vision component . . . 203

10.3.2 Agent Capabilities . . . 207

10.3.3 Discussion . . . 208

10.4 Conclusion . . . 209

11 Experimental Design, Implementation and Results 211

11.1 Overview and Hypotheses . . . 211

11.1.1 Generation of main hypothesis . . . 212

11.1.2 Definition of terms . . . 213

11.1.3 Chapter Structure . . . 214

11.2 Experiment Design . . . 215

11.2.1 Experimental Variables . . . 215

11.2.2 Supporting Hypotheses . . . 216

11.2.3 Experimental Stages . . . 218

11.3 Experiment Implementation . . . 219

11.3.1 Experimental Setup . . . 220

11.3.2 Questionnaire Design . . . 222

11.3.3 Choice of piece . . . 223

11.3.4 Capabilities of the system in relation to the piece . . . 225

11.4 Analysis of Results . . . 228

11.4.1 Data capture and participants . . . 229

11.4.2 Initial data preprocessing . . . 230

11.4.3 Direct analysis of question data 231
11.4.4 Preliminary Factor Analysis 233
11.4.5 Comparison of Conditions 240
11.4.6 Power Analysis 242
11.5 Discussion and further work 242
11.5.1 Experimental Setup 243
11.5.2 Performance of the System 244
11.6 Conclusion 245

12 Conclusions and Future Work 247
12.1 Recap 247
12.2 Evaluation 248
12.2.1 Original Aims 251
12.2.2 Overall conclusion 254
12.3 Future Work 254
12.4 Contributions 257

A Map of MAMA source code 259

B Derivation of two agent value relations 261
B.1 Automatic Generation 263
B.2 Manual Derivation 263
B.2.1 $R_{prev} = \texttt{SAME}$ 263
B.2.2 $R_{prev} = \texttt{SUBSUMED}$ 263
B.2.3 $R_{prev} = \texttt{SUBSUMES}$ 264
B.2.4 $R_{prev} = \texttt{ALTER}$ 265
B.2.5 $R_{prev} = \texttt{DISJOINT}$ 265
B.3 Descriptions of Musical Actions 266
B.4 Example Musical Act Analysis 274

C In C 277
C.1 Complete Score 277
C.2 Coded Score 281

D Canto Ostinato 283
D.1 Excerpts from the score 283
D.2 Coded Score 290

E Example Questionnaire: Piano Duet Interaction Survey 293

E.1 Instructions 293

E.1.1 Playing Instructions 293

E.2 Initial Questions 294

E.2.1 Personal Information 294

E.2.2 Musical Competences 295

E.2.3 Playing with others 297

E.3 Excerpts 299

E.3.1 Interaction 1 299

E.4 Post Questions 302

Bibliography 305

List of Figures

1.1 Relations between components of the thesis 2

3.1 Jackendoff's decomposition of Musical Idioms 38
3.2 Continuator: the tree of the patterns found in $\{$A B C D$\}$ 50
3.3 Architectural overview of Andante system 52

4.1 Example configurations for musical middleware 58
4.2 Agent System Overview 61
4.3 Configurations for Musical Multi-Agent System 63
4.4 Two agent platforms and human participants collaborating 64
4.5 Overview of Environment functions 65
4.6 Agent system timing diagram 67
4.7 Timing points for the Environment to determine 68
4.8 Timing diagram for reactions to internal events 69
4.9 Timing diagram for reactions to external events 70
4.10 Contract between the agents and the environment 72
4.11 Single Agent Overview 72
4.12 Complete contract for musical agent system 77
4.13 Overview of music representation specification 79

6.1 Division of actions when playing music 92
6.2 Overview of the formal agent model 95
6.3 Lattice value operators 101
6.4 Illustration of different concept relations between an old value *o* and new value *n* 103
6.5 Chord Root Lattice 104
6.6 Partial lattice of sequences of chord roots 105
6.7 Partial lattice for chords with extensions up to the 7th 107

6.8 Two different lattices for analysing chords (showing chords related to C minor, Eb major and G minor) 108
6.9 Example music for analysis 109
6.10 Timeslice and state indexed descriptions of musical interactions . . . 113
6.11 Example of different common grounds for a group of agents playing jazz116
6.12 Available relationships for construction of a new value, with agents $a, b, \ldots, x$ from the point of view of b constructing a new value 118
6.13 Relations between the values of two agents a and b, from the point of view of a constructing a new value 118
6.14 Possible values for musical action signatures 121
6.15 Example of an action signature being extracted from playing 122
6.16 Example of two action signatures extracted from the playing of agents 124

7.1 Parts of the agent overview covered by the Implementation chapter . . 134
7.2 Overview of the music realisation process 135
7.3 Lifecycle of music representation 140
7.4 Object structure in MRA 142
7.5 Message flow in the Main Musical Loop 146
7.6 Musical Agent generation of music in response to messages 147
7.7 Visualisation of part of the score of Canto Ostinato using the MRA visualisation tools 150
7.8 AgentSpace: overview 151
7.9 AgentSpace: detail 152
7.10 Agent analyser screenshot 154
7.11 Agent reasoner screenshot 155

8.1 Parts of the agent system covered in Analysis and Generation Chapter 160
8.2 Analysis system overview 161
8.3 Annotating played notes with links to scored notes 162
8.4 Overview of rendering pipeline for producing music 165

9.1 Overview of the components covered in the Deliberation chapter . . . 172
9.2 State Transitions in Dialogue Formulation with Relations, with only novel values shown 174
9.3 Musical Action Signature State Diagram 174
9.4 Overview of the deliberative system architecture 176

9.5 Example numeric lattices 178
9.6 Combination of the two constraints on next value (nodes marked p are possible) 183
9.7 Visualising the actions extracted from playing 187

10.1 Agent Space explanation 197
10.2 AgentBox in use 204
10.3 Overview of the computer vision system used in the AgentBox 205

11.1 Variables involved in the measurement of system performance 215
11.2 Experimental setup block diagram 221
11.3 Questions used in questionnaire to probe *I* and *E* 224
11.4 Comparison of different system configurations 226
11.5 Visualisations of the correlation matrix for averaged question scores over the entire dataset; in both diagrams, green represents positive correlation and red negative 232
11.6 Average question scores and question variance for the entire dataset . 234
11.7 Visualisations of the explanatory power of different factors 235
11.8 Factor Loadings by question, for 2 and 3 factors, using Maximum Likelihood estimation, and varimax rotation 237
11.9 Summed absolute factor loadings by question group 238
11.10 Distribution of factor scores for baseline conditions (for 3 factor varimax factor analysis) 239
11.11 Distribution of factor scores for all conditions (for 3 factor varimax factor analysis) 240
11.12 Distribution of factor scores for target conditions (for 3 factor varimax factor analysis) 241

A.1 Map of MAMA source code 260

B.1 Relations between the values of two agents *a* and *b*, from the point of view of *a* constructing a new value 261
B.2 Possible Relationships produced by Prolog program 262

Chapter 1

Introduction

This thesis is concerned with the creation of interactive musical agents — virtual musicians who can improvise music — and how they become equal partners to human musicians. Current systems in this area lack certain features which are necessary for a high quality musician. In particular, they lack the ability to structure their output in terms of a communicative dialogue, and reason about their relations to their partners. The aim of this thesis is to develop a theory which models musical interaction and communication, and implement it as a musical multi-agent system. This theory and system will:

- develop a model of musical interaction, which is computationally implementable, communication oriented, supports musical agents in reasoning about the actions of others, and is suited to real-time musical applications.
- develop a method by which the effect of adding an implementation of this theory to a musical system can be quantitatively tested.
- develop the idea of intelligent musical agents to create musical avatars, assist with net-based composition and improvisation and offer novel ways of interacting with music.
- produce music, in real time, by interacting and communicating with human musicians.

In order to address these issues, Musical Act Theory (MAT) is developed. This is a novel theory, which models the use of performative actions in music, analogous to Speech Act Theory in linguistics. This allows a dialogue oriented analysis of music, and an exploration of communication in the context of musical performance. The

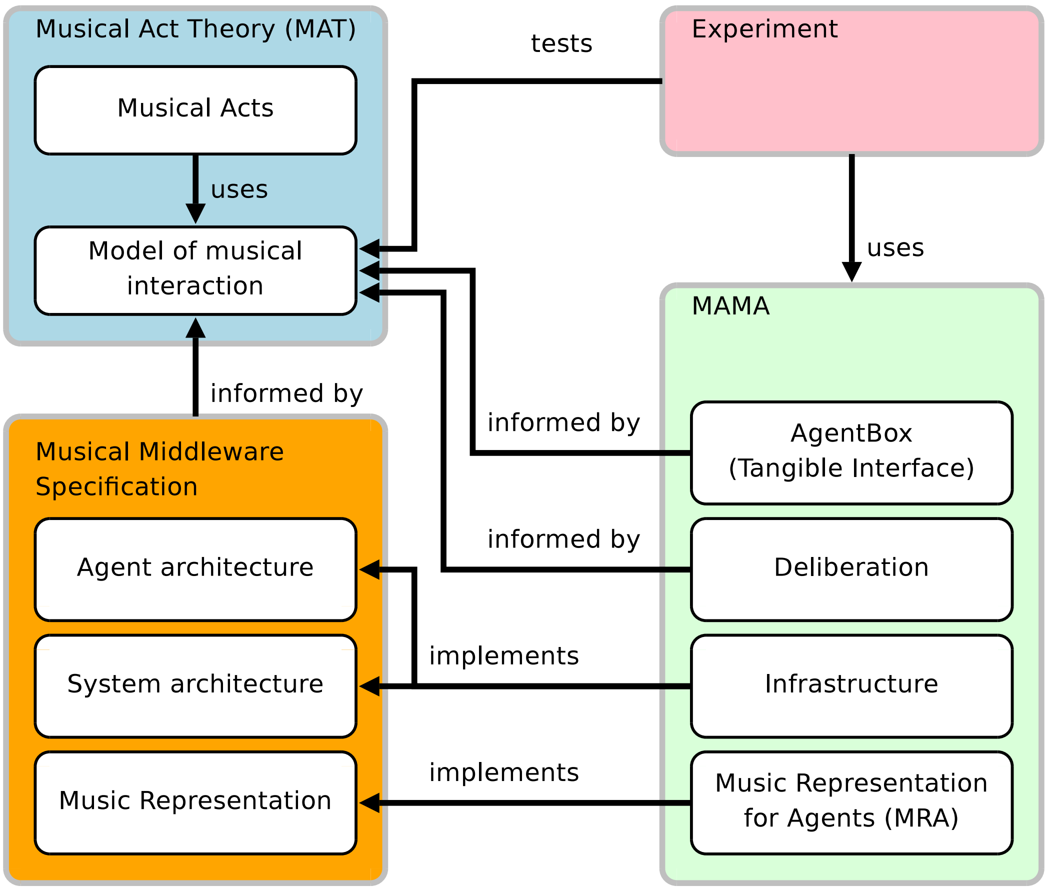


Figure 1.1: Relations between components of the thesis

work can be separated into four strongly linked threads — a specification for Musical Middleware, Musical Act Theory, MAMA — a complete musical multi-agent system based on the Musical Middleware specification and Musical Act Theory — and an experiment investigating the real-world functioning of this system and theory. In more detail (see Figure 1.1 for a graphical summary):

Musical Middleware The architectural centre of the thesis is a specification for Musical Middleware — a new term, which covers the use of musical systems to intelligently support a variety of music making applications. A variety of desirable configurations are presented, and in reaction to these, the following aspects of Musical Middleware are developed:

Agent Architecture gives a high level specification for the internal structure which a musical agent should implement in order to be part of a Musical Middleware system.

System Architecture describes the contracts between musical agents and the rest of the system which enable low-level musical interaction.

Music Representation defines an extensible music representation system for use by networked musical agents as part of the middleware system.

Musical Act Theory (MAT) In order to support the architectural decisions made about Musical Middleware, Musical Act Theory explores the issues involved in musical interactions from a theoretical point of view. MAT is inspired by Speech Act Theory, and its use in Multi-Agent Systems, and offers a computational, logic based framework for analysing and generating music. The central principle of MAT is that we can abstract from the musical surface produced by each musician a series of discrete actions.

Musicians are modelled as agents, who have the power to perceive, interpret and generate music, with each agent interpreting music it perceives according to its own capabilities and stylistic background. There are two parts to this theory:

Musical Acts are performative actions which are carried out by musicians, through their playing as part of a group. Identification and generation of these acts is the long term goal of this work, which the rest of the thesis lays the groundwork for.

Model of musical interaction is a bottom-up model which describes what happens when musical agents play music together. The low-level exchange of music between agents is formally defined using temporal logic, in a form suitable for a Musical Middleware system. The ability of agents to extract high level representations of music is then modelled with concept lattices to provide a formulation for agents to reason about the musical beliefs of themselves and others. Finally, based on these ideas, a complete set of discrete "Musical Actions"[1] is developed, which provides a substrate for agents to construct actions in response to those of others, by modelling a musical interaction as a stream of discrete, interrelated actions carried out by the agents involved, through their playing.

[1]Musical Acts and Musical Actions are different terms; Musical Acts are *intentional, performative* actions carried out through music, while Musical Actions describe changes to the musical surface in a generalised manner. This thesis is inspired by the idea of Musical Acts, but only deals with Musical Actions

MAMA The MAMA system is created in order to reify the ideas of MAT and Musical Middleware. It is a functioning musical agent system, which works in real time and can perform in several of the configurations suggested by the development of Musical Middleware. This consists of several components:

Infrastructure is a complete implementation of the system and agent architectures defined in the Musical Middleware specification. It contains mechanisms for exchanging music between agents, tools to run, manage and interact with agents, basic agent classes which can be used to build musical agents with a variety of capabilities, a set of classes for extracting and representing high level features of music along with classes which extract these features from musical performance and apply them to musical scores to create performances.

Music Representation for Agents (MRA) is a full implementation of the representation specification from Musical Middleware. It consists of a text based representation language, a parser for this language and a set of classes to represent all the objects in the language.

Deliberation is a set of functionality which uses Musical Actions to shape music in response to human musicians. It consists of:

- a set of symbolic versions of the musical features provided by the infrastructure which are compatible with the model of musical interaction.
- methods to extract these symbolic features from human playing and turn them into Musical Actions.
- a simple machine learning system which learns sequences of these Musical Actions and uses this database to choose Musical Actions to enact when playing with human musicians.

As part of MAMA, several case studies were created, to explore different aspects of the system:

In C demonstrates the system functioning as a Musical Middleware "Installation", where high level extramusical human input is used to influence the performance of this seminal minimalist piece by a group of musical agents.

Canto Ostinato uses the deliberative system developed to assist in playing piano duets in a musical manner, shaping the music in response to expressive features extracted from human playing.

AgentBox is a tangible interface, created to explore alternative ways to interact with the agent system. It uses a computer vision system to track the position of physical counters which represent musical agents, and integrates this with the Musical Middleware system. Overall, it provides an interface for people to influence the behaviour of the agent system in an understandable but non-trivial manner.

Experiment An experimental design is constructed which allows the interactivity of musical agents to be compared, as well as exploring the functioning of the MAMA system under controlled conditions; it is based on human musicians playing duets with the musical system, and using a questionnaire to rate their experiences. A pilot of this design is run, where pianists play with unseen partners, which include other humans, recordings of humans and MAMA in varying configurations.

To sum up, the contributions made are:

- a specification for a Musical Middleware system, which can be implemented computationally, and allows distributed agents to collaborate on music in real-time.
- a computational model of musical interaction, which allows musical agents to analyse the playing of others as part of a communicative process, and formalises the workings of the Musical Middleware system.
- MAMA, a musical agent system which embodies this theory, and which can function in a variety of Musical Middleware applications.
- a pilot experiment which explores the use of MAMA and the utility of MAT under controlled conditions.

1.1 Structure of the Thesis

The structure of the thesis is as follows:

1: Introduction sets out the primary motivation for the thesis, and outlines the claims and contributions it makes

2: Background presents an overview of the fields which are relevant to this work, and some tools and methodologies for assessing and classifying musical system.

3: Related Work describes a selection of theories of music analysis and improvisation, and a variety of musical systems which are relevant to the current system.

4: Architecture and Design This section starts by discussing situations where musical agents can be used, and coins the term "Musical Middleware" to describe the functioning of intelligent, distributed agents playing music alongside human partners. From these, and from other work, an architecture for musical agents is created, which allows agents with a variety of capabilities to work together. This architecture is used through the rest of the thesis.

5: Musical Acts describes a top-down view on intentional actions in music, and relates this to the musical actions developed in the previous chapter.

6: Theory of Musical Interaction develops the theory of musical actions; it starts from a formal description of a musical interaction, builds up higher level musical features, and then discusses the relations between the features of different players over time, constructing the final model of musical interaction.

7: Implementation describes the implementation of the agent system; it covers the creation of musical agents, the manner in which they exchange music in real time over a network, and the description language used to represent the musical surface.

8: Analysis and Generation in MAMA covers the manner in which musical agents analyse music over a set of features, and use the same feature set to generate music.

9: Musical Deliberation covers the construction of a reasoning system based on MAT. A theoretical framework is developed, which is then implemented as a functioning part of the system. The reasoner extracts musical acts from the features explored in the previous section, deliberates over a response, and then returns a set of features to be embodied in the output of the agent. Musical Acts are extracted from human playing to create a set of data to train the reasoner.

10: Case Studies presents three case studies, where the system has been used to perform a certain task. In order, they are the encoding of *In C*[2] for performance by a group of virtual musicians; addition of a tangible interface to allow human users to shape the playing of the system for *In C*; encoding of sections of *Canto Ostinato*[3] to allow the system to play piano duets in real time, responding to the musical output of human partners.

11: Experimental Design, Implementation and Results describes a pilot experiment which is used to test the operation of the system. Human participants are asked to play piano duets with an unseen partner, which they rate using a questionnaire; factor analysis is used to recover a score for the interactivity of the partner in each duet. Five conditions are analysed, with three providing baseline scores, allowing a comparison between two conditions of interest: the system reasoning about musical acts, and the system attempting to mimic the playing of its partner as closely as possible. The results are encouraging, although no significant conclusions are drawn from this pilot.

12: Further Work and Conclusions draws some conclusions about the system as a whole, and ties together conclusions about the individual components. Some directions in which the theory could be taken, and areas to which it could be applied are outlined.

[2]a minimalist piece by Terry Riley
[3]a minimalist piece by Simeon ten Holt

1.2 Glossary

This glossary contains terms which are either novel to this thesis, used in a non-standard manner or which the reader is expected to be unfamiliar.

Communicative Act is used in the same sense as FIPA Specification, an action which has an *intention* to communicate [Searle, 1983, page 165]. It is used as a substitute for Speech Act, where an action is being carried out, but the medium is not natural language speech.

Fragment is a small amount of music, which occurs within a single timeslice.

Facet is some aspect of the the musical surface which may be analysed — for instance, extracting chords or classifying rhythms.

MAMA is the multi-agent system which is created as part of this thesis.

MRA is "Musical Representation for Agents", the music representation specification and implementation created in this thesis.

Musical Action is a discrete action embodied in the musical surface.

Musical Act is an intentional, performative act, embodied in the musical surface.

Musical Surface is a representation of music at the level of discrete sound-objects, e.g. notes, in the sense of Lerdahl and Jackendoff [1983].

Timeslice is a span of time.

Chapter 2

Background

This chapter covers the research which is contextual to the thesis; specifically the studies of musical activities, models of the music making process, the linguistic theories which are influences and some background on multi-agent systems

2.1 Introduction

By its nature, this thesis touches on a wide range of disciplines; music is a primarily human activity, so background from psychology is necessary; creating a system which understands humans requires input from cognitive science; many computational musical systems fall under the remit of artificial intelligence; and this work draws on ideas from linguistics and speech act theory. This chapter serves to introduce an overview of work in these areas which is relevant to this thesis, and is structured as follows:

- some musical activities are introduced, so that terms can be defined for later use, and literature relevant to those activities is discussed.

- a range of models of music and improvisation are presented, so that the current state of modelling of musical activities can be discussed.

- a brief overview is given of concepts in Pragmatics and Speech Act Theory which are relevant to this thesis.

- a brief overview of multi-agent systems is given.

- some classifications for computer music systems are discusses.

Together, these discussions should ensure that the general area which this work is directed at is clear, and that most general concepts used in the rest of the work are familiar.

2.2 Musical activities and their computational, cognitive and psychological friends

This section explores the different activities which go into music making and how they relate to each other and to established programmes of research. There are many musical activities which can be, and have been, modelled. One way of dividing these into general categories is:

Listening and Analysis The transformation of music-as-perceived into abstracted representations

Performing Transforming a score (of some description) into some form of audio (however indirectly). This may include aspects of improvisation, and certainly includes interpretation.

Composing Creation of a musical score for later performance

Improvising Determining aspects of music to be played during performance

These are rough descriptions, and need further elaboration; also, many musical activities will draw on more than one of these modalities simultaneously. The following section will attempt both formal and informal definitions of the activities under analysis, and then talk about the related fields in psychology, cognitive science and informatics as appropriate.

2.2.1 Listening and Analysis

Listening, in the most general sense, is the abstraction of information from a stream of music; similarly, musical analysis seeks to create structures from a low level representation of music. The most obvious form it takes is when acoustic signals enter the ear of a human being, and have some effect on cognition. However, there are more situations which we would like to address using the term listening, such as:

- a computer program taking audio input and extracting some information from it;
- a computer program which takes some symbolic representation of music and extracts information from it;

The commonality here is the abstraction of information from a source of music; the music may be symbolic or acoustic, the processing may be done in real time or “offline” and the degree of information may be large or small, but the act of listening remains a process of generating some form of model, representation or extra information about the music being listened to. Musical perception is a skill which must be learnt by humans, and the creation of computational systems with similar skill is an active research area.

An early and pervasive formalisation of this idea is Schenkerian Analysis, which can be used to reduce pieces of music to an *Ursatz*, or fundamental structure; however, no mechanical method is given for arriving at structurings. A pedagogical introduction to Schenkerian Analysis can be found in Forte and Gilbert [1982]. This style of analysis is not universally accepted - see for example Narmour [1977, 1990].

In a similar vein, Lerdahl and Jackendoff’s Generative Theory of Tonal Music (see Section 3.1.2) has been called a listening grammar [Jackendoff, 1987]. Another similarity which all three systems share is a difficulty with computational implementation, as they rely on rules which, while accessible to humans, are not formally expressed to the extent necessary for algorithmic embodiment.

As an example of a method which does not share this issue, Dannenberg has also developed a model of music listening which recovers underlying structure from audio representations of music based on pattern detection - see Dannenberg [2002], Dannenberg and Hu [2002] for details.

2.2.2 Performing

In a very simple sense, performing music means taking a high level representation, and transforming it into an acoustic waveform, in front of listeners. There are aspects of this that are made less clear by the introduction of computers into musical performance:

- does the performance need to be real time?
- does playing back a tape piece count as a performance?
- what is the minimum amount of transformation required to produce a valid performance?

There is, implicit in the commonly accepted notion of performance, the idea of *expressive performance*. This is the *interpretation* of a musical score, by a *performer* in order to create a stream of sound which is felt to be expressive:

> What makes a piece of music come alive ... is the art of *music interpretation*, that is, the artist's understanding of the structure and 'meaning' of a piece of music, and his/her ... expression of this understanding via *expressive performance* [Widmer, 2001]

See Widmer [2001], Widmer and Goebl [2004], Ramirez and Hazan [2005], Juslin [2003]. Here, the object is either to take some form of score (or abstract musical representation), and turn it into a more concrete version, which is regarded by humans as being "expressive", or study the mechanisms by which humans accomplish this. Psychology studies how it is that humans create expressive performances, and what makes them expressive, while informatics is interested mainly in creating systems which can perform music expressively. However, it should be noted that both disciplines often have a relaxed attitude to the setting and "liveness" of performance. Much of the psychological data is collected in the laboratory (although some work with recordings of famous performers), and many of the computational models do not work in real time.

Exactly what is meant by expressive is the subject of some debate. However, Juslin [2003] defines five processes which provide the expressive character of a performance:

Generative Rules are used by the performer to indicate structure in the music. This approach is explained by [Clarke, 1988], backed up by evidence from [Gabrielsson, 1987], [Sloboda, 1983], [Palmer, 1996], and reviewed in [Clarke, 1995] among others. Variation in timing, dynamics and articulation are used by a performer to clarify group boundaries, metrical accents and harmonic structure. This can also be seen in the work of Widmer [2001], and the commentary from Jackendoff on the GTTM -

> The difference between a mechanical performance and a musically satisfying one lies in the performer's understanding of the role of the individual notes not just as elements in a sequence but in building integrated structures. [Jackendoff, 1987, page 235]

Emotional Expression allows a performance to portray certain moods or emotions - see Juslin [2001] for more details.

Random Variability due to the lack of determinism in the human motor-system.

Motion Principles are the dynamic patterns associated with human movement. This covers both the shaping of music to relate to patterns of human music (e.g. final ritardandi) and the patterns which arise from the interaction of a musician's physical body with an instrument

Stylistic Unexpectedness can be used by a performer to violate expectations, and serve aesthetic functions.

The psychological field of expressive performance analysis is wide, so we will not attempt a full exploration here. However, a large amount of information can be gathered from reviews by Gabrielsson [Gabrielsson, 1999, 2003]. The first reviews over 500 papers, while the latter adds a further 200, and a tabulation is given to keep track of the numbers of papers in different subdomains.

In terms of expressive performance systems, the review by Widmer and Goebl [2004] gives a clear overview of the current state of computational modelling of expressive performance. Four models in particular are looked at[1]:

KTH The KTH model [Friberg et al., 2000] is a rule based system, determining dynamics, timing and articulation, and using a local music context. This model is unique in using an “analysis-by-synthesis” approach, where new rules are evaluated by a professional musician in collaboration with the researcher

Todd Model This model [Todd, 1992] assumes a strong link between musical structure and performance, and attempts to model this using a simple ruler. Despite lacking some level of power, it has been used to investigate the “residuals” — details of individual performances.

Mazzola Model Taking a different approach, Mazzola [2002] has created a relatively self-contained mathematical theory of musical analysis and performance, which gives significantly different results to established systems. Unfortunately, no hard data on the quality of the expressive performance is available.

Machine Learning is the approach taken by Widmer (and also Ramirez and Hazan [2005] among others). Large amounts of performance data are analysed, in order to learn rules for note placement, phrase shapes etc. Since these rules have been extracted from real data, the output of the rules can then be compared with the rest of the corpus.

It should be noted that these rules all operate essentially on the note level, and particularly with piano based works, so that note lengths, timings and dynamics determine almost all of the performance. Systems such as Ramirez and Hazan [2005] and Arcos and de Mántaras [2001] deal with lower level expressive features, such as the

[1]only the most recent citations are given here - refer to [Widmer and Goebl, 2004] for further reading

spectral characteristics and dynamics curves of notes, as they work with saxophone performances. SaxEx [Arcos and de Mántaras, 2001] is particularly interesting, as it combines case based reasoning with the GTTM, while Ramirez and Hazan [2005] use a genetic algorithm approach to machine learning.

2.2.3 Composing

The traditional view of musical composition (in the western musical canon) is that of a composer creating a work, which is represented by a score. This score is an ideal representation of the composer's wishes, to be faithfully followed by performers. This may prove to be a limited concept; instead, a piece may be said to be composed of certain elements which are determined ahead of time, and certain elements which are determined in the moment.

Ed Sarath defines composition as

> …the discontinuous process of creation and interaction (usually through notation) of musical ideas. In other words, the composer generates materials in one time frame and encodes the work in another [Sarath, 1996, page 2]

This gives the composer the ability to work within the timescape of the composition, but also to take a broad view and examine the whole, or focus on smaller segments as necessary. Benson, on the other hand [Benson, 2003, pages 25] talks about a "true" composition having two key qualities - premeditation and permanence[2].

2.2.4 Improvising

Improvisation is to some extent a harder activity to quantify than the others; it is shrouded in mystique, and has had less formal analysis than other forms of musicality. However, since it is the area in which the current project is rooted, some time will be spent exploring just what improvisation really is.

Paul Berliner's seminal study of jazz improvisation begins with the following (oft-repeated) quote:

> I used to think, How could jazz musicians pick notes out of thin air? I had no idea of the knowledge it took. It was like magic to me at the time. — Calvin Hill

[2]this should not be taken to represent his full view, which is richer and more important, and will be explored later

This conveys a popular perception of improvisation as an impenetrable skill, allowed to some talented individuals and based very much on instantaneous intuition, inspiration and insight. While these three qualities are undeniably important for improvisation, there is far more in the way of hard work, practice, learnt structures and interrelations than implied by the Hill's original viewpoint. The rest of section will detail two views on improvisatory practise, structured around several questions. These views are given in: Benson's "The Improvisation of Musical Dialogue" [Benson, 2003] and Ed Sarath's "A new look at improvisation" [Sarath, 1996]. We will explore these views, keeping in mind the following questions as points of contrast:

- How does improvisation relate to performance?
- How does improvisation relate to composition?
- What are the temporal aspects of improvisation?

Benson's Views Benson's views stem from philosophical work, influenced largely by the continental tradition, and he presents a "Phenomenology of music"; a relation between empirical observations of musical practice which is consistent with philosophical and musical theory. He starts his characterisation of improvisation by relating it to both performance and composition. Composition is characterised as "designating or selecting musical features", while performance is "putting into action those features", and the "traditional" view of improvisation is that it relates to construction *ex nihilo*, "without sketches, manuscript or memory" (pages 23–24). Improvisation is seen to be similar to extemporaneous composition, in that it is not an interpretation or re-presentation of a work; something new is being created, with a unique identity, as opposed to the idea of an underlying work which allows many performances of the same idea. However, unlike composition, there is no premeditated or decided character, supported by a quote from Stravinsky that a musical work is "the fruit of study, reasoning, and calculation that imply exactly the converse of improvisation"[Stravinsky, 1970, page 138][3].

There is also a lack of permanence associated with improvisations; until the advent of recording technology, they were largely transient phenomena. Now that many improvisations are recorded, a large difference is that a composition is *prescriptive*, and details what should be done in order to play a piece, while a recording of an improvisation is *descriptive*, and details what happened on one particular occasion.

[3]Stravinsky's view on this matter is clearly not in line with the current project

The solution to this apparent problem of characterising improvisation in relation to both performance and composition is to talk of the improvisational parts of both of these activities; composers never create ex nihilo, but rather improvise, whether on existing material, or more broadly on the tradition in which they work. Similarly, in any performance, there is some degree of improvisation - some degree of judgement made in real-time about how to execute certain features. Attention is also drawn to Godlovitch's "Musical Performance" [Godlovitch, 1998], which characterises scores as "frameworks, like story lines, scenarios, or scripts awaiting completion through collaboration by players. . . " (page 82), and that there are no differences between improvisation and performance "so utterly as to make them stand in radically different relations to the music made".

In order to illustrate this view, Benson develops a series of (nominally) graded levels of improvisation [Benson, 2003, pages 26–30]. These will be covered in more depth as an axis for classification of musical systems 2.5.5, but a few of these levels are:

Improvisation$_1$ "Filling-in" certain details that are not notated in the score, e.g. tempi, timbre, attack, dynamics.

Improvisation$_2$ Addition of notes to the score that the performer is *expected* to perform, such as trills, and filling in figured bass parts.[4]

Improvisation$_5$ Addition or subtraction of complete measures, passages or scores

Improvisation$_{10}$ The composer uses a particular work as a template, to produce a more complex (or simply different) work.

By illustrating the spectrum of improvisatory practises, Benson emphasises the lack of distinction between improvisation and the other musical activities; improvisation is not a subset of either performance or composition, rather an activity which permeates all musical undertakings.

Finally, one compelling notion is that of a musical work as a space to dwell (after Heidegger [1963]). As a part of dwelling in the space, use is made of the surroundings, and they are in turn transformed, creating a dynamic relationship between works, performers, composers and listeners. To finish:

[4]Improvisation$_{1,2}$ are both common in Baroque scores, forming an expected part of Baroque musical practice.

> ...[A musical work] provides a world in which music making can take place. Performers, listeners and even composers in effect dwell within the world it creates. And their way of dwelling is best characterized as "improvisation" ...[Benson, 2003, page 32]

Sarath's Views Sarath speaks from the viewpoint of a practising musician, music teacher and theorist, seeking to explain what happens when one improvises. He draws on ideas such as implication-realization theory [Narmour, 1990] and temporal nonlinearity to construct a picture of improvisation as separated from composition by the improviser's attitude to temporality.

In order to explore this, three forms of temporality are defined[5]:

Expanding temporality is the chief mode of conductors; here, events in time have cumulative effects, and each event is built on the aggregate of all previous events. The composer can "freeze" time, and traverse past events at will. Implications for new events are based on the entire structure of what has gone before, and access may be given to what is coming next. This conception also *expands* towards both the future and the present.

Inner-directed or "vertical" temporality is the basic state of the improviser; events in sequence are only dependant on the previous event. In this Markovian mode, the artist creates moment to moment, with each event being self-contained and autonomous, following a continuous movement towards the localised present [6].

Retensive-Protensive borrows aspects of both expanding and inner-directed temporalities; the improviser projects awareness both into the past and future, but is still working within the present-directed framework of vertical conception. This is used in improvisation to recall past ideas, and have a sense of length and overall shape for the improvisation; it is used in composition to allow the working out of ideas through spontaneous performance; it is the basis of extemporaneous composition, where a work is produced in a single, real-time attempt.

These temporalities are not mutually exclusive; rather, different modes of working draw more heavily on one or another, and different activities can be said to have a pri-

[5]Some of the terms and uses of language may not be immediately apparent from the account here, but the general meanings should be clear

[6]the localised present is characterised as dealing with discrete events and their realisation, as opposed to the "overarching present", which deals with the present as part of the past-present-future sequence, subsumed within an eternal sense of presence

mary and subordinate conception in effect at the same time. So improvisation is characterised as primarily inner directed and subordinatively retensive-protensive, while composition is primarily expanding and subordinately retensive-protensive. This allows Sarath to make a distinction between improvisation and extemporaneous composition, as the latter is primarily retensive-protensively and subordinately inner-directed - a reversal of dominance from improvisation.

Improvisation then works as a series of inward and outward movements; in the inward phase, the improviser looks within to *internal imagery* to find possible new directions (*actualities*) for continuation. In the outward phase, the implications of these actualities are generated, and one is expressed. Finally, the probability conceptions may be neutralised - this means that the current action may be replaced by a new one, a new cycle in the inward-outward process. The faster these event cycles take place, the more responsive the resulting improvisation will be — if a several bar continuation is conceived and then executed, no new directions are taken within that time span. However, if the rest of the probability tendency is neutralised by the improviser before the end of the action, a new, more appropriate continuation can be made. This is key in interacting with others - the musical situation changes constantly, and one must be able to react quickly in order to be open to impulses from the rest of the group.

The use of a “referent” is also covered - some underlying format which gives a framework for musicians to work within. This contrasts with the general quest of neutralizing commitments to future events, by placing constraints on what a player must be doing when. The combination of both moment-to-moment and past-present-future structures is found in retensive-protensive temporality, allowing the improviser to work at a local level, but with awareness of the surrounding structures. Referents do not specify the entirety of what is to be played, either - the improviser may “deconstruct” the referent to find alternative realisations or treatments, depending on the concreteness of the referent. This leads to a continuum of activities (and referents) from total improvisation, starting with no explicit material through to musics with strict compositional content - “even in works entirely composed, performers will have some degree of creative options through volume, dynamics, inflection . . . and other expressive nuances”. In many improvised formats, there is also a cyclical nature to the referent. This tends to move towards a vertical conception of time, and allows another avenue towards inner-directedness within a structured framework.

Benson and Sarath, as presented here offer two very different views of what improvisation is; where Benson’s improvisation is centred around the notion of *works*, and seeks to situate improvisation in relation to styles and schools of music, Sarath describes improvisation as a dynamic process, unfolding over time, related but distinct from composition. Both of these viewpoints are influences for the treatment of improvisation which is developed through the rest of this thesis.

2.3 Linguistic Theories - Pragmatics and Speech Act Theory

This section covers Pragmatics and Speech Act Theory, linguistic tools which have formed the starting point for Musical Acts. Pragmatics is the branch of linguistics, or more generally semiotics, which is concerned with the relations between signs and interpreters. One definition is due to Morris [1938], who divided semiotics into three categories:

Syntactics The formal relation of signs to one another

Semantics The relations of signs to the objects to which the signs are applicable

Pragmatics The relation of signs to interpreters

Alternatively, from Stalnaker [1970]:

> Syntax studies sentences, semantics studies propositions. Pragmatics is the study of linguistic acts and the contexts in which they are performed.

Levinson [1983] argues that a formal definition of pragmatics is difficult:

> It is difficult to formulate a single definition and theory of Pragmatics; the selection of a pragmatic theory must to some extent depend on the choice of *semantic* theory, and there is no single homogeneous semantic theory.

However, pragmatics touches on areas such as deixis, conversational implicature and presuppositions. One of the important points made by pragmaticists is that the import of utterances (sentences uttered in a context) may be entirely different from their literal meaning. Grice [1957] introduces the form $meaning_{NN}$ to capture the idea of the *nonnatural* meaning of utterances, formalised in [Levinson, 1983] as:

> S *meant-nn z* by uttering U iff:
>
> 1. S intended U to cause some effect z in recipient H
> 2. S intended (1) be achieved simply by H recognizing intention (1)

In other words, S intends to cause z by getting H to recognise that S intends z.

It is these qualities, of being concerned with the *performance of acts in context*, and considering the *import* of an utterance as disjoint from its literal meaning, which make pragmatics an appealing area of theory to apply to music.

2.3.1 Grice's Maxims

An important part of pragmatic theory is Grice's theory of conventional implicature. Grice's "Cooperative Principle" [Grice, 1975] says:

> Make your conversational contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged.

Maxim of Quantity "relates to the quantity of information that is to be provided"

1. Make your contribution as informative as is required
2. Do not make your contribution more informative than is required

Maxim of Quality "try to make your contribution be one that is true"

1. Do not say what you believe to be false
2. Do not say that for which you lack adequate evidence

Maxim of Relation "Be relevant to the conversation"

Maxim of Manner "relates to how something is said - be perspicuous"

1. Avoid obscurity of expression
2. Avoid ambiguity
3. Be brief (avoid unnecessary prolixity)
4. Be orderly

This is (or has been interpreted as) applicable to communication in general, not simply spoken conversation. A particularly relevant application is in [Coventry and Blackwell, 1994], where the authors use Gricean maxims to analyse the difference between "cool" and "bebop" solos:

> Simply saying that Dizzy Gillespie (a bebop player) plays more notes than Miles (in his cool period) is uninformative. It does not get at the intentionality of the player, and does not give insight into the choice of notes that are played.

2.3.2 Speech Act Theory

The theory of speech acts is based on the works of Austin [Austin, 1962] and Wittgenstein, but is largely attributed (especially in Computer Science oriented areas) to Searle [1969]. It can be seen as a part of pragmatics, more specifically part of Dialogue Analysis. The defining idea is that when someone speaks, they are performing an *action*, and doing so with *intention*. This is partly in response to the verifiability criterion of meaning of the Logical Positivists, which discounts statements with no computable truth value [Neurath, 1996, page 38], and has problems with statements such as:

- I bet you sixpence that it will rain tomorrow
- I apologise
- I (hereby) declare war on Zanzibar
- I christen this ship the H.M.S Flounder

These are all statements which cannot be regarded as true or false, but have the form of an action.

Searle systematises Austin's theory, and gives a hierarchal taxonomy for verbal events:

Utterance Any verbal expression. Does not necessarily convey any meaning at all. A ”pure” utterance could be a nonsense rhyme used while skipping etc.

Propositional Utterance A propositional utterance has a reference to some real or imagined thing. ”A blue hamster” would be a propositional utterance. It refers to a thing, and meaning can be shared between the speaker and the hearer if they share a verbal code.

Illocutionary Utterance If a propositional utterance is intended to make contact with a listener, it becomes illocutionary. The intentional nature of the statement is paramount. The same statement can be caused by different intentions, in different contexts, and hold different meanings.

Perlocutionary Utterance If an illocutionary act is attempting to cause a change in the world (or at least the hearer's actions) they become perlocutionary.

Here, *loquor* means speaking, so *ill*ocution means “in the locution” and *per*locution means “through the locution”

He defines three types of condition for an utterance to be a Speech Act:

Preparatory The necessary conditions for the Act (e.g. to christen a ship, there must be an unchristened ship there, the speaker must have the authority to christen it etc.)

Sincerity The speaker must believe the effect the act will have

Essential The definition of the Act.

And further, he gives some categories of Speech Acts:

Directives Ordering, requesting, advising

Commissives Promising, threatening

Expressives Thanking, apologising, welcoming

Declarations Declaring war, christening, marrying

Representatives Stating, concluding

An alternative set of necessary conditions for modelling a dialogue using Speech Acts (in [Levinson, 1983]) is:

- There are unit acts (speech acts or moves) performed in speaking which belong to a specifiable, delimited set.

- Utterances are segmentable into parts, each of which corresponds to at least one act.

- There is a specifiable function to map utterance units into speech acts and vice versa.

- Conversational sequences are primarily regulated by a set of sequencing rules stated over speech act types.

2.3.2.1 Speech Acts in Multi-Agent Systems

This idea of performatives has been widely used in communication between computational agents in multi-agent systems, particularly as evidenced by the FIPA[7] Communicative Acts Library, and KQML [KQML Spec]. The FIPA Communicative Acts Library Specification [FIPA Specification] considers a message to be a combination of a performative and some logical content. 22 types of performative are listed, including Accept, Inform, Request, Not Understood and Refuse. There are four basic types given - Inform, Request, Confirm and Disconfirm. These performative acts can then be combined into larger complex protocols. As well as defining the names of these acts, definite semantics are given, for when they may be used, and what agents may believe about the world after their use. For example, INFORM has the following semantics for agent i informing agent j of proposition ϕ:

Model:	`<i, inform (j, ϕ)>`
Feasibility Preconditions:	$B_i\phi \wedge \neg B_i(Bif_j\phi \vee Uif_j\phi)$
Rational Effect:	$B_j\phi$

In natural language this means that:

- the act has the form of a message stating that i informs j of ϕ.
- in order to for i to send such a message, i must:
  - believe ϕ.
  - not believe that j has existing beliefs about ϕ.
- once i has sent the message, j will believe ϕ.

By specifying communicative actions in this manner, agents may reason about the beliefs of other agents, and complex patterns of interaction may be set up separately from their content; FIPA provides several protocols, such as Contract Net, English Auction, Recruiting and Subscription.

2.3.3 Relations between music and language

There have been many works which apply linguistic tools to the analysis or creation of music. Grammars are a particular favourite, with Steedman [1996] using a transformational grammar to model blues chord sequences, and the Bol Processor [Kippen

[7]Foundation for Intelligent Physical Agents

and Bel, 1992, Bel, 1998] using grammars to represent tabla playing knowledge. As previously mentioned, Coventry and Blackwell [1994] use Grice's maxims to explore the different qualities of trumpet solos. Walker [1997] provides an in-depth analysis of jazz playing using conversational structures. Lerdahl and Jackendoff [1983] should also be mentioned here - it is detailed more thoroughly in Section 3.1.2

Finally, Beghilli [1995] gives a specific application of Speech Act Theory to music; looking at Verdian opera, he finds stylised, iconic gestures, which through their association to certain situations may be used to *perform* actions musically. In the opera Stiffelio, for example, the orchestra carries out the act of weeping for one of the characters - rather than supporting the actions of the character, the orchestra actually *enacts* the weeping. There are a set of emblems in Verdian opera which represent performative acts, and since each emblem only covers certain facets of the musical surface, they can be applied in many contexts.

2.4 Multiagent systems

The study of multiagent systems is an endeavour which requires a certain amount of positioning, since it is informed by many disciplines but is a coherent field in its own right. A good introduction to the field can be found in Wooldridge [2001], Weiss [1999]. Within the field, there are many different directions and methodologies, but the main division [Wooldridge, 2001, page 7] is between using agent systems as a paradigm for software engineering, and as a means to model and understand social behaviour. In this context, what is meant by an agent, and by extension a multi agent system? Wooldridge [2001, page 23] suggests that an agent must be:

Reactive to its perception of its environment,

Proactive in attempting to fulfil its goals,

Socially able to interact with other agents, and possibly humans.

Other qualities attributed to software agents include [Weiss, 1999]: autonomous, goal-directed, intelligent, distributed, decentralized, asynchronous. However, different projects generally pick and choose which of these qualities are most important to their particular goals.

Agents typically communicate with other agents using an Agent Communication Language (ACL) such as [FIPA Specification] or [KQML Spec], both of which are organised around the use of communicative actions with clearly defined semantics, which allows the use of predefined protocols to manage interaction.

2.5 Tools and frameworks for classifying and evaluating computer musical systems

This section examines several taxonomies or methods of classifying and evaluating musical systems, in order to develop axes for comparing musical systems, and situate the current work with reference to similar systems.

2.5.1 Expressive Completeness and Structural Generality

In Wiggins et al. [1993], the authors develop a framework for evaluating musical representation systems. Looking at three different situations - recording, analysis and generation - they use two axes:

Expressive completeness (EC) is the ability of a system to represent “raw” data - the range of musical events which can be captured.

Structural Generality (SG) is the range of high level structures which can be modelled and manipulated in the representation.

A distinction is made between scoring systems (which represent models of how a piece should be played) and representations of musical objects (which represent a performance of a piece), and the report is mostly concerned with the latter, although some systems - particularly grammars - are as suited to generation as description. In fact, several of the systems are moving towards scores or programmes for music, and the axes used are equally applicable.

Some examples:

MIDI scores low on both EC: it can only encode pitch, duration and indications for timbre and dynamics, and also for structural generality: there are no high level features available.

Bol Grammars score low for EC as they only represent symbolic tabla strikes, but higher for SG, as high level patterns can be represented.

CHARM Harris et al. [1991] uses abstract data types to represent music, and as such, scores highly on both counts, as it can be extended to deal with whatever information is required (subject to certain constraints, such as constant pitches and mathematical relationships for pitch and time).

These terms provide a useful, but not necessarily complete means of analysis for deciding what capabilities of a system are important. Wiggins et al. [1993] applies them to descriptions of musical objects, but it is clear they can be applied to some extent to scores as well. One may analyse existing musical works, and discuss what capabilities in both of these directions are necessary to represent them - for example:

In C (Terry Riley) requires little expressive completeness (as no instrumentation or dynamics are explicitly given) but a medium degree of structural generality in order to represent it

Stimmung (Stockhausen) requires high degrees of both expressive completeness and structural generality to represent the overtone singing which is used and the underdetermined route through the piece itself

2.5.2 Interactive Systems - Scores, Responses and Players

Rowe [1993, pages 6–8] describes three axes on which to classify musical systems, with the aim being "to recognize similarities between them and to be able to identify the relations between new systems and their predecessors."

The three axes described are:

Drive is the relation of the system to predetermined events; *performance-driven* systems do not have an expectation of the music they expect to find at the input, while *score-driven* systems do.

Response covers how the system creates music. It may be:

- **Transformative,** where existing material undergoes transformation; this material need not be stored, however - it may appear at the input of the program.
- **Generative,** where rules are used to produce complete musical output from some fundamental material or knowledge
- **Sequenced,** where existing fragments of music are output, with minor alterations (e.g. tempo and dynamics).

Agency describes the system's relation to its "player"; *instrument* systems elaborate on human input, but a piece played on one would be considered a solo, while *player* systems tend towards a musical presence, and a performance would feel more like a duet.

(the naming of these axes has been added to facilitate future discussion)

2.5.3 Net Music Approaches

Net Music, or music based on Interconnected Musical Networks is the field of musical systems which use technological networks to allow players to interact musically. These interactions are generally real-time, and can overcome physical boundaries to create new musical social spaces. In [Weinberg, 2002], Weinberg gives an overview of the field, and then discusses a taxonomy of Net Music approaches. These are:

Servers allow players to communicate with the server, but not with each other. The server may provide musical material for the players to work with, but inter-player communication is non-existent.

Bridges connect players, so they can play as if they were in the same space - they virtually simulate a connected physical space. The system does not try to enhance the interaction, but provide as close to a "natural" interaction as possible.

Shapers allow players to influence the output of a central server. Although the players can hear the effects of other players, they cannot directly communicate with each other.

Construction Kits allow diachronic collaboration over musical material; participants can create their own material, and modify that of others, and then place the results on a communal server for further modification.

These terms are designed to describe some of the new musical approaches which are emerging. They are not complete, and systems may have aspects of several, but they do suggest an axis on which to analyse, evaluate or situate work. In effect, this provides a discussion point for two distinct qualities - temporality and communication method. If the temporality of the interaction between users and the system (and hence other users) may be synchronic or diachronic and communication (between users) may be absent, directly through music or through manipulation of shared musical objects, we arrive at Table 2.1. This also touches on the idea of music as a *coordination artifact* [Viroli and Ricci, 2004, Omicini et al., 2006]

	Temporality	
Communication	Synchronic	Diachronic
None	-	Server
Direct	Bridge	-
Shared Objects	Shaper	Construction Kit

Table 2.1: NetMusic approaches organised by communication type and temporality

2.5.4 Methodological Approaches and Motivation

There are many different motivations for involving computers in music; the motivation for the development of a particular system will both shape the capabilities of the system, and determine the ways in which it can be evaluated. For a similar discussion regarding AI in general, see Bundy [1990]

In order to characterise systems, it is necessary to look at the intention behind the system so that the results of the work can be properly characterised. Since the history of computer supported music is nearly as long as the history of electrical computers themselves, the field of computer music has had time to evolve and explore many possible reasons for its existence. However, this does not mean that the proponents of the genre have always been clear, in their minds or in their communications, about the reasons to construct a particular system or investigate a certain area.

Motivated by understanding these concerns, and addressing the "stagnation in the body of published work involved in the development of computer programs which compose music", Pearce et al. [2002] develop four types of computer music programs, explaining their motivations and methodology. These are paraphrased as follows:

Algorithmic Composition covers programs written to extend personal compositional practice. The *motivation* here is artistic: supporting the composer in creating pieces which would otherwise not have been created. *Methodologically*, the only constraint on this work, or evaluation which may be carried out is the composer's aesthetic judgement.

Compositional Tools are general tools to aid any composer in composition. The *motivation* is to make available to others the results of research into computer music, and present different approaches as part of a toolkit for composers to draw on. This work can be assessed with reference to software development *methodology*:

analysis, design, implementation and testing phases are all necessary, and have their own evaluation criteria.

Theories of Musical Style attempt to create programs which are computational models of stylistically valid music. The *motivation* is to allow the empirical testing of models of musical style, by generating pieces using the models. *Methodologically* this approach allows assumptions to be made explicit, and compositions generated by the model may be compared to existing examples so that the model can be checked for over- and under-generation

Cognitive Theories of the processes supporting compositional expertise can be modelled using AI techniques. The *motivation* here avoids aesthetics and stylistic validity, and is solely concerned with understanding the underlying cognitive processes involved in human composition. The *methodology* is to:

- state hypotheses embodied in the model.
- derive these hypotheses from *psychological experiments*.
- evaluate the hypotheses through attempts to refute them based on the output of the model.

These four types of compositional tool, along with their own evaluation methods, give a clear path towards creating clear and incisive assessments of musical projects; however, in order to assess the current work, some points should be addressed:

- not all work falls exactly into one of these categories; When presented with a categorisation scheme, such as outlined above, it is often possible to find works which belong to several categories, or to none. Two responses to this are:
  - The classification scheme is too constrictive, and must be relaxed to allow works to belong to several categories
  - The work was poorly conceived, and had it been situated cleanly in one category would have been stronger

  The second point can prove useful in the design of computer music projects - unless there is a clear reason for working cross-category, it should be avoided. The first point provides a richer toolkit; if a project falls into several categories, then appropriate parts of the project may be analysed according to the different categories. This is a notion we will use extensively in the analysis of our own work, as there are clearly aspects to be evaluated as several categories. We will also attempt to defend the creation of a multi-category work.

- these categories deal specifically with *composition*, and none of the other musical activities characterised previously. This means that some of the methods will not be appropriate for this work, but where possible, a similar spirit will be adopted.

- when looking at musical tools, arguably the most important criterion is ecological — does the tool become widely used, or result in interesting music which would not have happened otherwise? These are questions which can only be answered after some period of time has passed.

2.5.5 Types of Improvisation

Improvisation is often thought of as an activity in its own right, separate from the activities of performance or composition. However, a more flexible approach is to view all performance oriented activities as having some improvisational content – by discussing what characterises the improvisation involved, we gain an understanding of the capabilities needed for that particular style of performance.

[Benson, 2003, pages 26-30] gives a series of increasingly “free” types of improvisation. All of these acknowledge some form of referent, although in the later forms the referent is not necessarily the piece being composed or performed. A sample of these is:

I_1 The players fill in certain details which are not specified by the score

I_2 The players add notes to the score, in a manner expected by the composer (e.g. trills)

$I_{3,5}$ The players add whole measures, sections etc. to the score; if this is in a manner approved or expected by the conductor, it is (I_3), otherwise, it is (I_5).

$I_{7,8}$ The score is changed considerably, reharmonization, melodic alteration etc. The difference between I_7 and I_8 is that of recognisability - in the former, there is some obvious connection between the score and the rendition, but in the latter there is not.

I$_9$ The composer uses a particular form or style of music as a template for composition - improvising on the form.

I$_{11}$ The composer and performer are part of a musical tradition, and by their work they modify the rules and expectations associated with that tradition.

If this spectrum is applied to computational musical systems, it gives a tool to discuss the capabilities of the system, and hence what kinds of improvisation it may be able to perform.

2.5.6 Creativity

Boden [1998] presents a taxonomy of creativity, directed towards analysis of artificially intelligent systems. Firstly, there is a division between *P* creativity, which is creativity relative to that particular agent or system, and *H* creativity, which is for output which is historically novel — that is, has never occurred before. The second division is the *type* of creativity, as follows:

Combinatorial creativity relies on combining existing elements in novel ways; for example, analogies, where a relation is made between distinct concepts.

Exploratory creativity involves the exploration of structured spaces to generate novel concepts; this can generate surprising ideas, but they will be coherent with the existing space or body of work.

Transformational creativity alters one or more dimensions of the space, so that ideas are possible which previously were not.

These give a framework for looking at the different ways in which artificial systems can be said to be "creative", and Boden [1998] gives several examples of different systems and the type of creativity they are capable of.

2.5.7 Conclusions

This chapter has introduced background material which will help to understand the rest of the thesis; a set of musical activities and our understandings of them are used to define the work being done here; Speech Act Theory and a general sense of pragmatics provide an inspiration for the system of musical communciation developed; ideas from multi-agent systems, in particular FIPA, are used as a template for the formalisation of communicative processes. Finally, some methods have been presented for classifying and evaluating musical and creative systems which will be used to evaluate the current work.

Chapter 3

Related Work

This chapter presents some theoretical models of different aspects of music, and some implemented musical systems, with some discussion about how the current work relates to these systems.

This thesis is concerned with two different approaches to the area of computational music making: theoretical and practical. The related work presented here hence covers a selection of theoretical models of different aspects of music, and then goes on to describe a range of different practical musical systems.

3.1 Models of music, improvisation, interaction and creativity

Several models are given in this section, which cover very different aspects of music making. After each model is presented, a short discussion about the strengths and weaknesses of the model is given, and at the end of the section, a final discussion points out the areas which are open to further research.

3.1.1 A cognitive model of improvisation (Pressing)

Formal models of improvised music are few and far between; possibly the most commonly cited is due to Jeff Pressing [Pressing, 1988]. Since this model has not been superseded in the literature, it will be explained in some detail, and then commented on.

Pressing is seeking to present a model for the cognitive processes which underly the generation of improvised music; hence, his model should both account for observed features of improvised music, and have a degree of cognitive validity.

This model works at the level of “musical events”; although these are not further defined, we assume them to be roughly analogous to notes, although the definition may be relaxed somewhat from the musical surface used in [Lerdahl and Jackendoff, 1983]. These notes are divided into “event clusters” (E_i), with each event being assigned to only one cluster. A complete improvisation is then written as:

$$I = \{E_1, E_2 \ldots E_n\}$$

For a given improviser, we then add:

Referent - R A piece specific guide or scheme - some form of score. The nature of this is left purposefully open (Pressing [1984])

Goals - $\mathscr{G}$ The goals of the improviser

Memory - M The improviser’s long-term memory

Put together, the process of solo event-cluster generation is:

$$(\{E\}, R, \mathscr{G}, M)_i \rightarrow E_{i+1}$$

while group playing adds C to represent the kth performer’s cognitive representation of the other performer’s previous output:

$$(\{E\}, C, R, \mathscr{G}, M)_{i_k} \rightarrow E_{i_k+1}, k = 1 \ldots K$$

In order to produce new output, during the interval (t_i, t_{i+1}) a series of steps is taken:

1. the output E_i (decided on during the previous interval) is triggered
2. E_i is decomposed into Objects, Features and Processes (more on these later). This works both on the intended output, and the actual output as it becomes available.
3. E_{i+1} is produced, based upon
   - long term factors - R, $\mathscr{G}$, stylistic norms and ongoing processes
   - evaluation of the effects and possibilities of E_i

   This can use *associative generation* or *interrupt generation*, and generates factors which give rise to E_{i+1}.

4. Cognitive and motor subprograms are set in motion, which will realise E_{i+1} - the beginning of the next iteration.

Musical events are represented under three distinct analytical representations (which I will shorten to “OFP arrays”):

Objects are unified cognitive or perceptual entities. They are represented as a category label (e.g. note, rest, glissando, scale) for each object, and a cognitive strength for that object

Features represent properties of each of these objects, along with the cognitive strength of each property. For instance a note may have features relating to its pitch, duration, dynamic, harmonic function etc.

Processes denote the processes which gave rise to the notes. They are represented by the name of a process, and a set of parameters and corresponding cognitive strengths for that process.

These OFP arrays are produced for several axes simultaneously; the three representations are constructed for acoustic, musical, movement and “other” aspects of E_k. There is a purposeful degree of redundancy here, both across axes (e.g. the performer would know that a particular motor programme will give rise to a particular sound) and within axes (e.g. representing a chord as DFAB or B*b* diminished 7th). This is so as to allow maximal flexibility of the choice of path through this space, by increasing the richness of connections between specific points. In terms of output, the redundancy indicates that event production is heterarchical, with different options being allowed precedence as necessary. The objects and features which one can perceive are assumed to be stable over the course of an improvisation - learning happens elsewhere.

This covers the infrastructure of the model, but the most relevant aspect is the way in which E_{k+1} is generated from these OFP arrays. In *similar associative generation*, successive event clusters have mostly similar values for their parameters. In *contrastive associative generation*, most strong parameters have the same values, but one or more of the strong parameters moves from one end of its spectrum to the other. However, in *interrupt generation*, all of the strong array components are reset without regard to their current values. Hence an improvisation may be modelled as a series of groups of event clusters, with the clusters within a group being produced by associative generation, and the start of each group being produced by interrupt generation. This can be modelled by simulating “boredom” using a time-dependant tolerance level and a repetition metric.

3.1.1.1 Discussion

Some important questions remain open here:

Finished output How is finished output created from the OFP arrays? Although a representation is given to use, there are many ways in which the component parts could be put together; some of these would depend on the architecture of the improviser, and some on the goals in force at that particular time. For more ideas in this direction, Pachet’s work on PACTs makes interesting reading [Pachet, 1994, Pachet et al., 1996].

Selection of continuations How is one possible continuation (OFP array for E_{i+1}) chosen over the others? There are many ways within associative generation that parameter settings can be modified for the next event cluster; why is one parameter chosen to take a contrasting value? How are the parameters of the “boredom threshold” chosen? Again, some of these may be due to the structure of the improviser, and any goals in place, but there is a deeper question, relating to creativity and individuality here; musicians will make *choices* about which structures to vary and how, and this is very much part of their *personality*. However, the fact that this model is non-prescriptive about the creative processes used makes it appealing as a general framework to refer to.

Multiparty improvisation The model specifies a variable C for the cognitive effects of the playing of others. This is clearly a vast simplification; there is no structure given, and no idea of how this is taken into account during the entire process. This is encouraging, since it leaves the question entirely open to investigation.

3.1.2 The Generative Theory of Tonal Music

In 1983, Lerdahl and Jackendoff published the Generative Theory of Tonal Music (GTTM) [Lerdahl and Jackendoff, 1983]. This was an attempt to provide a listening grammar for Western tonal music, with the idea that it could give an insight into more general music making as well.

Since the explanation of the GTTM is rather lengthy, it is necessary to state some reasons for interest in it, and in particular why such a full description of its workings is being presented:

- it remains the most widely cited model of musical cognition, and is certainly one of the most important in the field. However, this alone does not necessitate such an in depth exposition.
- it provided inspiration for previous work: [Murray-Rust, 2003, 2005]
- there are several concepts exemplified within it, which will be used in the work to come, notably:
  - the reduction hypothesis, giving a framework for further complexity/specificity of musical ideas;
  - the use of simultaneous, complex axes to describe the interpretation of a piece of music;
  - the fact it is based on building up a complex representation from heard music.

The GTTM aims to build up a representation of music along the lines of that which an expert listener would construct on hearing a piece; This is explained more thoroughly in Lerdahl [1988], through an exploration of the analyses of serialist works, particularly Boulez's *Le Marteau sans Maître*.

A serialist work is organised according to a comprehensive structure. However, when listening to the work, it is not always possible to discern this structure:

> . . . Yet nobody could figure out, much less hear, how the piece was serial.
> . . . in the interim, listeners made what sense they could of the piece in ways unrelated to its construction. [Lerdahl, 1988]

This points to a large gap between the compositional system, and the cognized result. Hence, Lerdahl proposes a "compositional grammar", and a "listening grammar". A compositional grammar produces some organisation of the inputs or specification [1], and a sequence of events - the score. Upon hearing the piece, a listener will attempt to infer a set of rules governing the events within it, which then leads to a structural description.

[1]the text does not expand on this

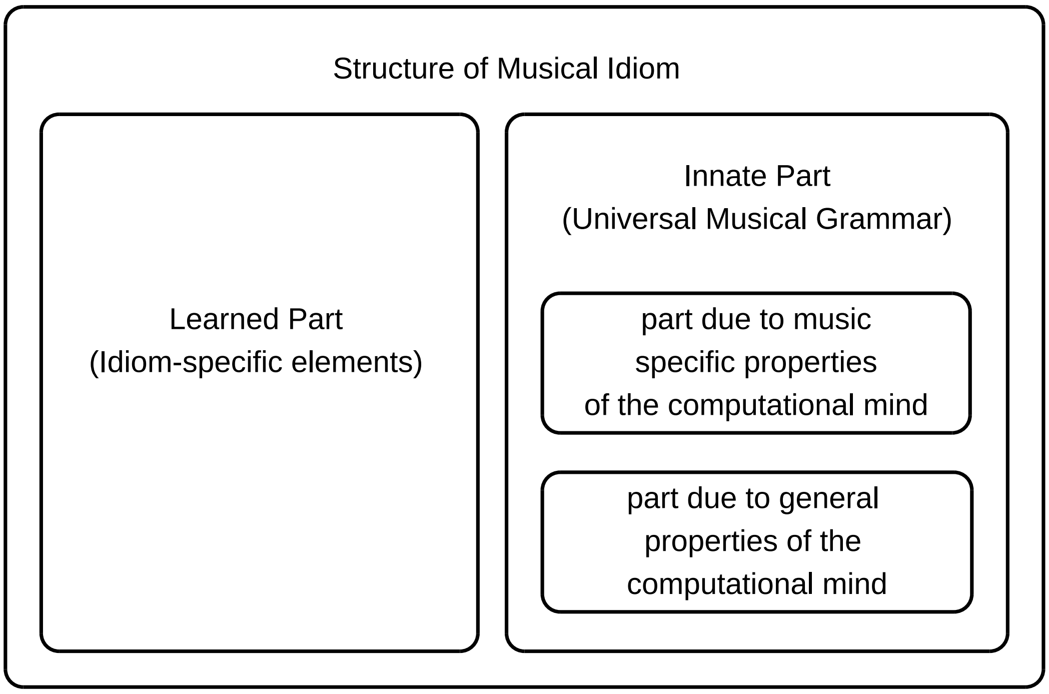

Structure of Musical Idiom
Learned Part
(Idiom-specific elements)
Innate Part
(Universal Musical Grammar)
part due to music specific properties of the computational mind
part due to general properties of the computational mind

Figure 3.1: Jackendoff’s decomposition of Musical Idioms

Lerdahl’s contention is that purposefully chosen grammars used in contemporary music are ‘artificial’ grammars - which do not *necessarily* relate to listening grammars. Historically, compositional grammars have been shaped both by artifice and ‘natural’ effects, so that listening and composition maintain relations; in the avant-garde, one is free to choose one’s own grammar, and the rift may begin.

This was Lerdahl’s motivation for developing the GTTM - to use a listening grammar to inform his creation of artificial grammars, which were “intellectually complex, yet spontaneously accessible”.

Ray Jackendoff’s motivation appears to come from a more cognitive angle; in Jackendoff [1987], he describes a series of models of various cognitive tasks; primarily natural language, but also vision and music. He describes each of these tasks in terms of structured levels, and explores their relations to each other. In the chapter on “Levels of Musical Structure”, he describes the GTTM in detail, and prefaces it by situating the cognitive abilities involved - see Figure 3.1.

The GTTM is not generative in the sense of being able to compose new pieces of music; rather it is “a set of principles that match pieces with their proper structures”

[Jackendoff, 1987][2]. It covers four levels, and for each level has a set of “well formedness rules”, which define allowable structures, and “preference rules” which indicate which structures should be preferred; there is a second division in their ruleset, between “formation rules”, which define the formation of a level in isolation, and “correspondence rules”, which deal with that level’s relation to other levels, including the musical surface.

The starting point of the GTTM is the “musical surface” - the encoding of music as discrete pitch-events. This musical surface is represented using Western music notation, but is assumed to be extracted from an audio signal: “it is easy to overlook the fact that the musical surface . . . comes to our perception only after a substantial amount of processing” [Jackendoff, 1987]

An important part of the GTTM is the “reduction hypothesis”, which is:

> The pitch-events of a piece are heard in a hierarchy of relative importance; structurally less important events are heard as ornamentations of elaborations of events of greater importance.

The four levels of the GTTM are as follows:

Grouping The surface is hierarchically decomposed into groups, which relate to motives, phrases and sections. Well formedness rules are used to ensure that a properly nested hierarchical structure is built up, while features such as proximity, similarity, symmetry and parallelism are used by preference rules to indicate preferred groupings.

Metrical Structure A metrical grid is imposed over the music which is heard. Each level consists of a series of dots, with each dot representing a beat at that level. This starts at the smallest metrical level, and progresses upwards with progressively fewer, but stronger beats on each level. Each layer of the grid is uniform in its spacing, at either 2 or 3 times the period of the layer below. This accounts for metrical regularity up to a few seconds in length, and hence does not deal with long term structure.

Time Span Reduction Working from the reduction hypothesis above, time span reduction works in the melodic and harmonic domains to allow events to be merged into more important neighbours, with the more important part being labelled the *head*, and the lesser the elaboration. This gives a tree view of the entire piece of music, according to the rhythmic-structural importance of pitches.

[2]The GTTM is generative in the same sense as Chomsky’s generative grammars

Prolongational Reduction Again working from the reduction hypothesis, and creating *domains* made from a *head* and an *elaboration*, prolongational reduction combines elements, but instead looks at patterns of tension and relaxation; three types of prolongation are given:

Strong Prolongation where the head and elaboration are the same chord, and the tension does not change

Progression where the head and elaboration are different chords, and the tension can increase (if the elaboration follows the head) or decrease (if the head follows the elaboration)

Weak Prolongation where the elaboration is a weaker (less stable) version of the head.

3.1.2.1 Discussion

This has been a brief tour of a complex model; some of the implications, limitations and insights due to the model are now discussed.

Firstly, the notion of the musical surface can become confusing. The authors state clearly that the musical surface is derived from music-as-heard, but represented in the text by Common Practice Notation. There is a significant issue here, in that a common practice notation representation of music does not indicate the actual positions and durations of notes - rather it gives a performer indications as to when and how notes should be played. In fact, the performer's interpretation of the underlying musical structure translated into decisions about when and how to play notes contributes largely to the domain of *expressive performance* - this can be seen in relation to Widmer's recent work [Widmer, 2001], which seeks to create expressive renditions of a piece based purely on the musical structure.

Secondly, the GTTM as a whole is directed solely at creating representations of music; it is a "listening grammar", rather than a compositional grammar. This does not mean it is useless for the work at hand, but rather that it can only be a part of the answer. For musical agents, it can provide insight both into interpreting the actions of the rest of the group, and how others will interpret one's own actions. This second sense seems very close to Lerdahl's stated intention of using a listening grammar to inform compositional grammars, keeping a tighter feedback between the abstract or formal structure of the music and the resulting representations in listener's minds.

Finally, although the GTTM is a cognitive theory, it is not readily computationally implementable; parts of the theory have been implemented computationally (e.g. Smoliar [1995]) but there has not been a total computational implementation of the entire GTTM. There are many aspects which need to be developed, two examples being models of parallelism and the weighting given to different preference rules.

3.1.3 Live Algorithms

The Live Algorithms for Music project [LAM] seeks to create algorithms which can take their place on stage alongside human musicians. A live algorithm should be (paraphrased from [LAM])

> interactive and autonomous; an ideas generator; idiosyncratic but comprehensible. A live algorithm can collaborate actively with human performers in real-time performance without a human operator. A live algorithm can make apt and creative contributions to the musical dimensions of sound, time and structure

The model underlying live algorithms is derived from a simple model of interaction [Blackwell and Young, 2004]. Two people (A and B) are playing together; A is outputting audio X, while B is outputting Y. In order for an interaction to be taking place, we need X to be dependent on Y, and vice versa, so A has:

- a set of perceptual functions, P, which create internal representations p from Y.
- a decision process, $F(h_a)$, which creates a set of musical representations q.
- a process Q which acts over the intermediate representations to produce audio.

So, the full process is:

$$Y \xrightarrow{P} p \xrightarrow{F(h)} q \xrightarrow{Q} X$$

3.1.3.1 Discussion

This gives a very broad framework with which to construct computational improvisers; they hear, interpret, process and realise. The philosophical thrust of the live algorithm movement goes further; desirable qualities of the system are set out (as noted above) with a constant emphasis on autonomy, creativity and surprise. This model allows for the fact that the improvising systems need not be human, or in any way "cognitively valid", and follows the idea that in assessing a live algorithm, "the performance is the laboratory". The issue of the generality of the theory, however, is that little guidance is given as to how music could and should be processed by live algorithms; while this is desirable from the point of view of constructing a range of systems, it can be problematic from the point of view of constructing systems which work well together, as there is no shared knowledge about how other systems are likely to behave.

3.1.4 A Framework for the Analysis of Performer Interactions

Pelz-Sherman [1998] presents an analysis framework for Western Improvised Contemporary Art Music (WICAM). This is developed using a phenomenological approach: a series of "micro-score" improvisations were carried out, and the recordings of these were analysed.

The theory starts from the notion of an intelligent musical agent, capable of producing and interpreting musical signals, and using its intelligence to generate new plans and alter its behaviour to optimise the performance of the group. The agents have the ability to:

> 1. make accurate judgements *in real time* about the "semantic intent" of each performer.
> 2. accurately convey the semantic intent of their own musical ideas to other performers *in real time*.
>
> [Pelz-Sherman, 1998, page 127]

Agency is defined as the ability to influence the character of the musical as perceived by the audience, and reflects the degree of autonomy and intention which the agent displays. Groups of agents whose playing is very similar are said to be "agent systems", to reduce analytical complexity.

The interaction between these agents is modelled in communication theory terms: each agent may be either *sending* or *receiving* musical information. Agents are sending when they play music which has a high rate of change, or a lot of "musical information"[3]. Alternatively, senders initiate musical ideas, while receivers respond to them.

When musical information is passed from one agent to another, an *I-event* occurs; these are distinct from solo events and preplanned actions, and make it clear to the listener that some form of interaction has taken place. A taxonomy of i-events is given:

[3]the exact nature of musical information is not defined, but the intent should be clear

Imitation events occur when some component of the music produced by one agent is adopted by another. This can include a large degree of transformation, but the effect for the listener is that one agent was imitating the novel output of another.

Question and Answer events differ in that the response need not have a direct relation to the call; there should be a consequential relation — so it sounds as though one bit of playing is in response to the other — but there need not be any musical relation between the two bits of playing.

Completion/Punctuation happens when one agent takes a predictable direction, and another agent completes the gesture (whether alone or in synchrony).

Interruption events involve one agent playing in a "directionless" manner, and being sharply interrupted by another; this provides a point of departure to start on a new musical idea.

Next, some modes interaction are described; between two agents, each can be in a state of sending, or receiving. If both agents are sending, this is *sharing*; if neither is sending, it is *not sharing*; if one is sending, that agent is *soloing*, while the other is *accompanying*. These can then be combined into two- and three-phase structures, for example sharing → soloing/accompanying is emerging/withdrawing, while the reverse transition is merging/accepting; the pattern soloing/accompanying → sharing → soloing/accompanying is interjecting/supporting and so on. When considering groups of more than two agents, each agent may be in a different mode with each of its peers, so complex modes of interaction may be constructed. Finally, modes may be present at different levels; for example, during a solo, the mode will be predominantly soloing/accompanying, but other modes may occur briefly.

From analysing further material using these techniques the idea of qualitative measurements of interactivity is introduced: perhaps looking at the frequency of i-events gives a good picture of the level of interactivity, or looking at the distribution of agency through the piece (i.e. the blocks of time for which performers were "sending").

3.1.4.1 Discussion

This is a highly relevant piece of work, as it presents a structured theory of improvisation which is strongly grounded in current musical practice. However, there are some limitations which allow scope for further work:

- the notion of exactly what constitutes an i-event is not clearly defined — at least, not clearly enough for a computational implementation. Similarly the notions of sending and receiving rely on a measure of musical complexity or novelty, and more work would be required to construct an implementation.
- the fact that this theory is strongly grounded in WICAM, while making it appealingly concrete, may limit the applicability to other areas; at least, it would require additional study.
- the range of possible modes and i-events seems quite small; both interruption and completion/punctuation are relatively "special cases", leaving imitation and question/answer as the main modes of interaction. It is my feeling that there are a large number of more subtle interactions taking place, which shape the interaction in a less obvious manner.

3.1.5 Other models of musical improvisation

There are several other theories of musical improvisation which should be mentioned here; Seddon [2005] describes six modes of communication used during jazz improvisation, with each mode being either verbal or non-verbal, and one of Instruction, Cooperation or Collaboration. Several situations are described, although a large amount of disagreement was encountered. Also, much of the discussion relates to instructions given *around* the playing.

Johnson-Laird [2002] gives an account of the relation between algorithms and creativity as applied to the construction of jazz solos by humans in real-time. Longuet-Higgins [1987] gives an interesting view on the relation between cognitive processes and music, while Mazzola [2002] gives a theory of everything musical including performance.

3.1.6 Discussion

In relation to the creation of computational musical agents, able to play with human performers as equals, there is a deficiency with all of these models. There is a need for a theory which is more developed with respect to output than Live Algorithms or Pressing's work; which is more concerned with performance than the GTTM; which is more generally applicable than Pelz-Sherman's work, and which is more computationally implementable than any of these, while also dealing with the complexities of several musical agents playing together.

3.2 A Selection of Computer Music Systems

In order to give a general overview of the current directions of computer music research, a selection of systems and approaches are presented here.

3.2.1 Constraint Satisfaction

Constraint Satisfaction is a technique used to solve combinatorial problems, by modelling a problem as a set of variables, with constraints over what values they may take. This has immediate similarities with music theory - for example, playing in a particular key is a constraint on which notes may be used and the rule of avoiding parallel fifths disallows certain arrangements of notes in chords. Pachet and Roy [2001] gives a good summary of the approaches taken so far, particularly focusing on traditional musical approaches such as four part harmony. [Truchet and Codognet, 2004] also details a series of musical problems using a CSP formalism, and creates solutions using adaptive search. Cope's EMI system (discussed in Section 3.3.1) depends extensively on constraint satisfaction.

A recent usage of constraint programming is the Strasheela system [Anders et al., 2004], a computer aided composition system, written in Oz - a language oriented towards constraint satisfaction. A composer creates music by imposing constraints on score objects. This set of rules is then run, to produce a score which fulfils the criteria, and can then performed (either by humans or sound synthesis).

Finally, Aucouturier and Pachet [2005] describes the use of constraint satisfaction for composing sequences from sampled drum sounds.

3.2.2 Generative, Evolutionary and Social Systems

There has long been an interest in creating music using generative and evolutionary algorithms. An early example of this is GenJam [Biles, 1999, 1998], which uses genetic algorithms to create jazz solos, and can "trade fours" with a human performer. Felice et al. [2002] describes a system based around genetic algorithms, which uses both formal metrics and human input over the web as fitness functions to evolve tunes.

Miranda [2002] looks at the way musical repertoires can evolve in a society of musical agents; these agents have apparatus for producing an perceiving sound, and interact by making sounds at each other. In the course of an interaction, agents attempt to model the model parameters that others use to make sounds, and respond with similar noises. Miranda [2003] continues in this direction, using cellular automata for synthesis and music composition. For other social and evolutionary approaches to music, see Bown [2006], Martins and Miranda [2007], Coutinho et al. [2005] among others.

3.2.3 Net Music

Net Music is an emerging field (or sub-field) which emphasises the use of the network, whether local or internet, in supporting music making. Summaries can be found in a special edition of Organised Sound [Landy, 2005] dedicated to the topic, and also [Àlvaro Barbosa, 2003]. A classification of these systems has been given in Section 2.5.3, and rather than discuss all the systems in detail, two systems are illustrated:

Firstly the CODES system [Miletto et al., 2005], which enables participants to cooperate in the construction of a “musical prototype” of a piece of music. “Lines” are made by choosing one of a set of patterns at each stage, with users able to create and modify patterns. The use of patterns allows untrained musicians to join in with the prototyping.

Secondly, the PIWeCS system [Whalley, 2004] is a marriage of intelligent agents, electroacoustic processing and network music. A variety of traditional Maori instruments were played, and loaded into MAX/MSP (a graphical dataflow language, with extensions for sound processing). Users can then control parameters of these samples via a web interface, to create an arrangement. To allow for users with a low skill level, agents would listen to the output of the users, and supply additional material, or enter into dialogue as appropriate.

Other systems and papers of interest are [Barbosa, 2005], which details a framework to allow collaborative compositions across the web, and [Kapur et al., 2005], which discusses the possibilities afforded by networked music making.

Finally, Open Sound Control [Wright, 2005] should be mentioned, both because it is a general enabling technology for networked music, and because it has been used in the current system.

3.2.4 Musical Companions

The term “Intelligent Musical Companion” was coined in Thom [2000], which presents “BoB”, or Band *out* of the Box — a tribute to Band in a Box, a commercial software package which creates static accompaniments for jazz and related styles. BoB is designed to be able to trade solos with a human musician in real time; it learns a particular user’s playing style in a warm-up session, and can then extract salient features of the human’s playing, and respond in a related manner as the performance progresses.

3.2.5 Intention-based systems

The COMPOzE system [Henz et al., 1996], and the related work in Zimmermann [1995b,a] explore the use of intentional specifications to guide computer systems in the composition of music. By combining constraint programming with an intention plan (relating to a multimedia presentation) background music can be composed which is structurally correct, and supports the form and mood of the presentation.

A contrasting exploration of intention in music is given in Gabrielsson and Juslin [1996], where a set of musicians were asked to attempt to convey different emotions through their playing, and listeners were asked to attempt to discern the performer’s emotional intention.

3.2.6 Musical Agent Systems

There are several systems which explore the ideas of musical agency, ranging from the philosophical to the software engineering uses of the term; Minsky [1986] mentions music several times (see also Minsky [1981]). Greussay [1985] offers a view of Beethoven which could be said to be agent oriented.

More recently, the Andante system [Ueda and Kon, 2003] provides a basic infrastructure for mobile musical agents — unfortunately, after a promising start, development seems to have ceased. Fonseka [2000] describes a musical agent system designed to play certain contemporary compositions. Agents run scripts, that define actions they take in response to events. While it is interesting from a technical point of view, it does not offer much insight into the musical process or the communicative aspects of musical interactions. Pachet [2000] uses groups of agents sharing a score to evolve rock and Batucada rhythms. Finally, Wulfhorst et al. [2003] creates a group of musical agents which can perform beat tracking and adapt chord harmonies to fit different progressions.

3.3 Three Related Works

This section presents a more in-depth look at three musical systems of particular relevance.

3.3.1 EMI

Cope [2001, 1996, 1991] describe Experiments in Musical Intelligence (EMI), which is a recombinant music composition system.

EMI composes a work using the following techniques:

- A database of music, stored as events, which are tuples representing onset, pitch, duration, MIDI channel and dynamics. The pieces are selected by a human to have a consistent style — composer, character, instrumentation, levels of ornamentation etc.
- segmentation processes, which break pieces into fragments representing measures, motives, harmonies etc.
- pattern recognition algorithms are used to extract the *signatures* of composers — patterns of notes which appear in some form in several of their works.
- the SPEAC model of functional relations gives a hierarchical description of music in terms of *statement, preparation, extension, antecedent and consequent* parts. It can be applied on several levels, and the musical fragments in the database are labelled with SPEAC codes to assist in choosing appropriate fragments.
- an Augmented Transition Network is used to choose correct harmonies and motives by generating sequences of SPEAC identifiers.
- a wide range of transformations and generalisations can be applied to the material in the database to allow it to be fitted to as many musical situations as possible, expanding the range of choice.

Putting all of these together, EMI produces pieces of music which are “in the style of” the input composers. EMI can either produce scores or performances, although human performances of the generated works are generally better regarded.

3.3.1.1 Discussion

The EMI system is included here because it is one of the most widely known and well regarded computational composition systems. As such, it has different goals to a real-time, improvising musical system. However, one of the most interesting aspects of EMI's output is its ability to evoke emotional responses in human listeners; Douglas Hofstadter, in the first chapter of Cope [2001] describes his reaction to the music produced by EMI as (occasionally) deeply emotional, containing human qualities which spoke to him. He found this troubling, as there was nothing in the system which seemed to relate to the meanings he was experiencing.

In relation to the prospect of creating virtual musicians, EMI gives a fine account of how existing styles can be analysed and replicated to create similar music. It does not, however look at real-time composition, or structuring its responses in reaction to human musical input.

3.3.2 Continuator

The Continuator is described in several papers, for example Pachet [2003, 2004b]. It is variously described as a flow machine, an interactive reflection system, a sequence continuator and an interactive musical system.

The Continuator is based around a Markov model of pitch event sequences, of the type produced by MIDI instruments - events have pitch, velocity, onset and offset times.

The model of musical structure used is a complete variable-order Markov model of input sequences [Pachet, 2003]. As events arrive, a *reduction function* is used to map from events to node-identifiers. Nodes are then added to the set of trees as follows:

- to add a sequence of nodes, each node except the last is added to in the tree, in right to left order.
- the *index* of the last node is then added as a possible completion to all of the nodes in this branch.
- each subsequence of the sequence is added; so when {A B C D} is added, the branches are: {A B C } completed with D, {A B} completed with C, and {A} completed with B - as shown in Figure 3.2.

In order to generate a continuation, the longest match for the input sequence is used to index a node in the tree, and the continuation is selected from those available at that

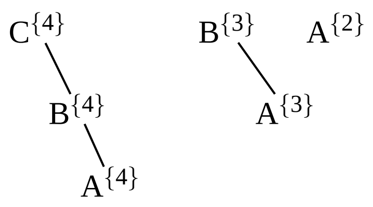


Figure 3.2: Continuator: the tree of the patterns found in $\{$A B C D $\}$

node; this selection is probabilistic, and biased by the number of times a particular completion occurred in the input data.

The use of reduction functions is a key part of the system; musical notes have several attributes, and attempting to represent all of these accurately would give a very large and sparse tree. Instead, a hierarchy of reduction functions is used, with increasing generality; pitch may be divided into regions rather than exact pitches, and other attributes may be ignored. So a suggested order of reduction functions is:

- pitch, duration and velocity
- small pitch region and velocity
- small pitch regions
- large pitch regions

This allows the tree to complete sequences without an exact match for the most descriptive reduction function, without introducing the computational complexity of Hidden Markov Models.

A range of strategies can then be applied to generate the rhythm for the notes, including using the rhythm which was present when the continuation was captured, imposing a linear rhythm on the notes and using the rhythm from the input sequence which is being continued.

A mechanism is used to assist the system in fitting in to the current musical context. Rather than simply choosing completion nodes based on the frequency of completions in the input data, a fitness function is introduced, which computes the level of fit of that node with information representing the musical context — for example the dynamics of the players, or the last few chords the pianist has played. A parameter S is used to control the balance between Markovian completion and reactive completion, so that:

- at $S = 0$, the system ignores the musical context, and completes from the probabilities in its database.

- at $S = 1$, the system ignores the probabilities in the database, and completes according to how well a node fits the current context.

Finally, [Pachet, 2004a] describes some additions to the Continuator which allow its use in different situations. The use of interaction protocols such as *turn-taking*, *single note accompaniment* and *phrase based accompaniment* provides a number of modes of interaction with the user.

3.3.2.1 Discussion

The Continuator is undeniably an impressive system; there are several reports of its use with both musicians and children: Pachet and Addessi [2004], Addessi and Pachet [2006], where it is seen to provoke “ah-hah” moments, encourage children in turn-taking, and sustain interest for more than a few minutes - an example of a musical Flow machine.

From the point of view of the current work, the areas of the Continuator which could be explored further are:

- the mode of operation is predominantly of playing in the style of the user; the system is compliant with the user’s wishes, rather than acting with agency to create novel responses to the user’s playing.
- while interaction protocols are used, they mostly cover *when* the Continuator plays — although some detail whether it plays phrases or chords — and do not influence the way in which it constructs responses.

3.3.3 Andante

Andante [Ueda and Kon, 2003] is an architecture for mobile musical agents: software agents who have the ability to make music, and also travel from one computer to another over the network. The desired capabilities for musical agents are:

Encapsulate an algorithm for the the composition of music; any extra input data required may also be carried by the agent.

Interact and exchange information with other agents, as human musicians do.

Interact with human musicians, either musically or through manipulation of agent parameters.

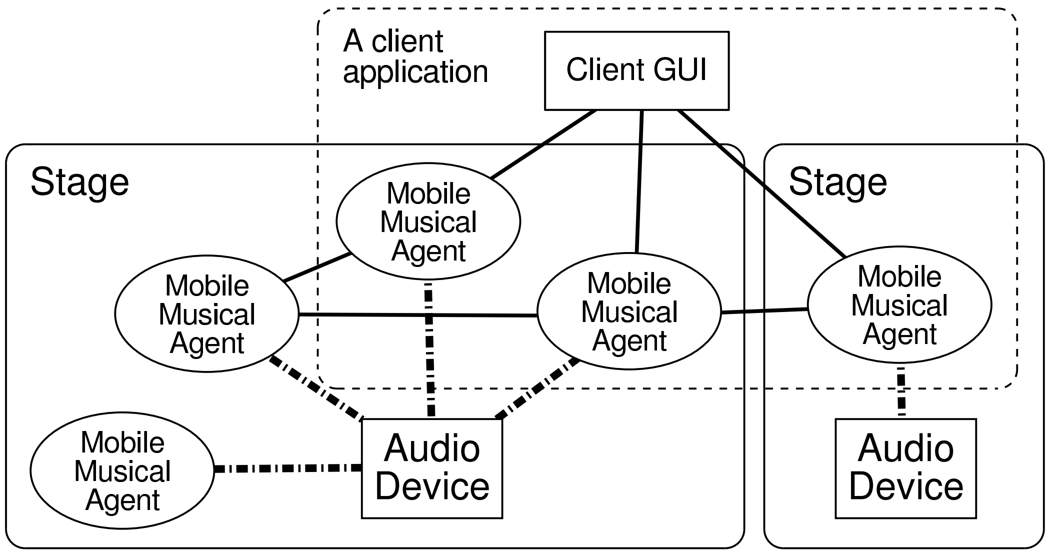


Figure 3.3: Architectural overview of Andante system

React to sensors which channel information about the real world into the agent system.

Migrate from one computer system to another.

The system is designed to be cross-platform and interoperable, so it is built in Java, using the Aglets mobile agent development platform. Figure 3.3 shows an overview of the Andante architecture. Each agent is executed on a computer which has a *Stage* — an environment which hosts agents and allows them to interact with each other. Each *Stage* also provides access to an audio device which the agents can use to produce music. A user interface can be provided which controls a set of agents who may be running on different *Stage*s. Each agent communicates directly with the audio device on the stage it is running on, and each stage provides a metronome to allow agents to synchronise with each other.

3.3.3.1 Discussion

Andante provides a particularly interesting feature — the mobility of agents. This means that (to use an example from the paper) if a system was set up in a museum, with a stage for each room, an agent could be assigned to each visitor, and track them around the museum, migrating from stage to stage as necessary. However a serious limitation of the system is that the agents have direct access to the audio system they are using. This means that while agents are mobile, they cannot communicate with agents on different stages, so interaction is only possible between local agents. Since the architecture does not provide mechanisms to deal with the latency of network communication, it cannot support a truly distributed group of agents making music together,

3.3.4 Positioning of the current work

Given all of this work, where should the current musical work be located? It can be seen as a synthesis of several of the approaches given here, with a constant emphasis on implementability and real-time operation; there is hence a space left for a system and a theory which:

- aims to produce high quality music, like the Continuator and EMI, but with a more developed notion of interaction and communication with human musicians.
- emphasises the social aspects of music making, but is more musically developed than the societies of noise-making agents discussed previously, with a clearer understanding of the communicative properties of music making.
- presents a model of musical improvisation which is more computationally implementable than Pressing's and Pelz-Sherman's work, and more suited to real-time musical applications than the GTTM.
- develops the idea of Intelligent Musical Companions and Live Algorithms, in tandem with ideas from Net Music to create distributed musical avatars for net-based composition and improvisation.

3.4 Conclusion

This Chapter has presented several related theoretical models and musical systems, and used these as a basis to suggest a direction and area for the rest of this thesis.

Chapter 4

Architecture and Design

This chapter proposes an architecture to be used for musical multi-agent systems, as a form of Musical Middleware. *This architecture is used throughout the rest of the thesis as the basis for design and implementation of the system.*

This chapter sets out the architecture which will be used to design and implement the musical agent system, and is structured as follows:

- the term *musical middleware* is introduced, and used as the guiding principle for creating a multiagent system.
- a model of the components which are necessary for the system as a whole to function is developed, with specifications for how the system is organised, and how it operates temporally.
- the internal architecture of musical agents is developed, based on the characteristics needed for musical middleware and creating high quality music interactively.
- the way in which music is represented in such a system is discussed, and a specification is given.

4.1 Musical Middleware

A starting point for the development of this musical agent system is the idea of allowing the computational agents to take care of as many of the tasks of producing music as possible, to allow human musicians to work with the high level aspects of music without having to deal with the minutiae. In software design, middleware often functions in this role, providing an intelligent layer between application components in a distributed system so that high level tasks can be carried out without reference to the underlying implementations.

The ObjectWeb Consortium [Krakowiak] gives the definition:

> In a distributed computing system, middleware is defined as the software layer that lies between the operating system and the applications on each site of the system.

and continues to give some important characteristics:

> - Hiding distribution, i.e. the fact that an application is usually made up of many interconnected parts running in distributed locations;
> - Hiding the heterogeneity of the various hardware components, operating systems and communication protocols;
> - Providing uniform, standard, high-level interfaces to the application developers and integrators, so that applications can be easily composed, reused, ported, and made to interoperate;
> - Supplying a set of common services to perform various general purpose functions, in order to avoid duplicating efforts and to facilitate collaboration between applications.
>
> These intermediate software layers have come to be known under the generic name of middleware.
>
> The role of middleware is to make application development easier, by providing common programming abstractions, by masking the heterogeneity and the distribution of the underlying hardware and operating systems, and by hiding low-level programming details.

With regard to making music, this can be viewed in two ways. Firstly, there is the traditional view, where the above comments are taken literally, and the middleware allows software written on different operating systems in different places to work together.

A different interpretation is that the musical surface is the basic substrate in which different agents work. By this view, the following interpretations of the properties can be given:

Heterogeneity A range of musical agents may work together; they might not all understand the same aspects of music, and they can certainly have radically different internals

Distribution The agents may be on different machines, in different locations

Abstraction The agents offer a means of higher level interaction to the user; the user does not have to work at the level of the musical surface.

Reusability Capabilities given to agents can be used in a wide range of situations; they can choose to apply techniques, or be instructed to apply techniques, as appropriate. The user can deal at the level of abstract behaviours, and not have to write any code.

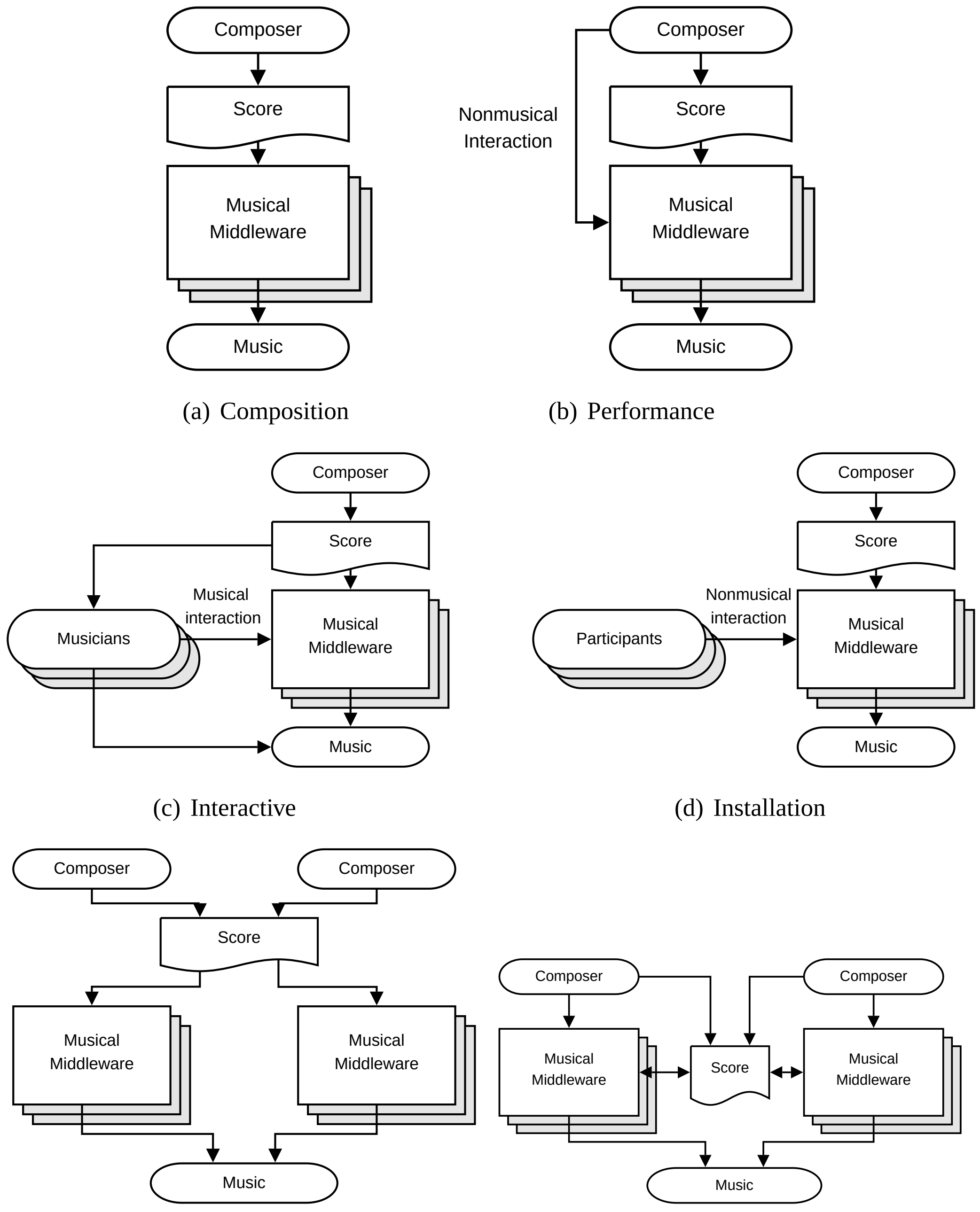
There are a variety of ways in which such a system could be used; some key examples are shown in Figure 4.1:

Composition (Figure 4.1(a)) A human user creates pieces of music at a high level, and the system can perform these pieces. The middleware allows the composer to create scores which are dynamic and nonlinear, and allows sections of the score to be highly underspecified, with the system intelligently filling in any gaps.

Performance (Figure 4.1(b) A composer creates a score, which allows room for degree of realtime input from the composer. The score is performed as a collaboration between the composer, giving high level direction and the middleware interpreting the direction in the context of the score. High level direction might be at the level of a band leader - calling for solos, reordering music on the fly etc. It could also explore new ways of working such as adding additional musical material or giving different agents (conflicting) tasks to accomplish.

Installation (Figure 4.1(d)) The system works from a score given by a composer, but the score allows it to be highly reactive to input from participants in the installation. This input could be in many forms; the positions of the users might be tracked, and the agents respond to this; the users might have buttons to press; there might be terminals people use to exert some form of high level control. Once the composer has created the score, the installation is autonomous, and needs no more “expert” interference.

Interactive (Figure 4.1(c)) Again, the system is given a score; in this instance, the score is then played in conjunction with a group of musicians, forming a hybrid ensemble. The musicians share a score with the system, and the system reacts to the output of the musicians.



(a) Composition (b) Performance (c) Interactive (d) Installation (e) Distributed (f) Distributed Jamming

Figure 4.1: Example configurations for musical middleware

Distributed Composition (Figure 4.1(e)) Two (or more) composers collaborate on a score. Each composer then passes the score onto their middleware system, which may be configured according to individual's tastes. In an extreme version, where several people collaborate, and each middleware system plays a single part in the resulting composition, we have a form of musical avatars.

Distributed Jamming Composers work on a minimal score, or pass some ideas on to their agents. The middleware systems then collaborate on a shared score, with each composer guiding their own system in the improvisation.

4.1.1 Motivation

The use of intelligent musical middleware has a variety of motivations, some social, some technical and some scientific. We can split these into two general categories - *enabling* and *analytical*.

4.1.1.1 Enabling Applications

Enabling motivations are situations which would be in some way facilitated by the technology. A clear enabling motivation is in distributed music making. It was seen in Section 3.2.3 that there is a lot of current interest in ”net music”, or ways to make music using the internet as a medium. This covers two key approaches:

- overcoming physical distance - simulating the closeness of geographically distinct participants.
- facilitating new means of collaboration.

Intelligent middleware has a lot to offer in the second application, such as:

- a common problem with network applications is latency. If a middleware system can take care of timing issues, then this eases the burden of latency; if a user is interacting with the system at a higher level, higher levels of latency may be permissible — if the system is taking care of the business of placing notes at the correct time, it may not be an issue if a request to “start playing the next section staccato” must be issued half a bar before it comes into effect.
- in many distributed systems, users cannot directly interact musically; this creates a need for a system which can process other types of input, and feed it into the creation of music.

- people build up online personas which can go some way to projecting their personality into the net. A musical middleware can allow users to create musical avatars: by altering the capabilities, knowledge and preferences of their system, they can create a personalised musical avatar, with the capability for persistent relationships to be built up.

4.1.1.2 Analytical Motivation

While the sounds which go into music may have important mathematical or physical relationships, it is the perception of music as music by humans that gives it its status. Many of the questions which can be addressed by intelligent musical systems are hence human questions; a group of humans playing music together may exhibit a wide range of capabilities, such as maintaining a unified tactus and predicting the actions of other participants. For an artificial musical system to function, it must display many of these characteristics; by creating such a system, we may gain insight into how humans carry out these tasks, or at least a greater understanding of what tasks they *do* carry out.

4.1.2 Capabilities

From looking at the diagrams of interaction situations (see Figure 4.1), some necessary capabilities of a Musical Middleware (MM) system may be determined:

Score All of the examples given above require some form of score - indeed, for any piece of music which is not entirely free improvisation, some form of score is necessary. There are many forms that a score may take, and it is clear that something more than a traditional linear Common Practice Notation score is necessary

Music Generation Technique Possibly *the* fundamental property of an MM system is the ability to output music. There are options here, though, as musical output may take many forms, ranging from audio signals to CPN scores, with many points in between.

Music Listening Just as a human musician, the system needs the ability to listen to the output of others, and in some sense “understand” their playing.

High level functions In order to function as middleware, the system must abstract some of the low level detail, so that users may deal with the system on a higher

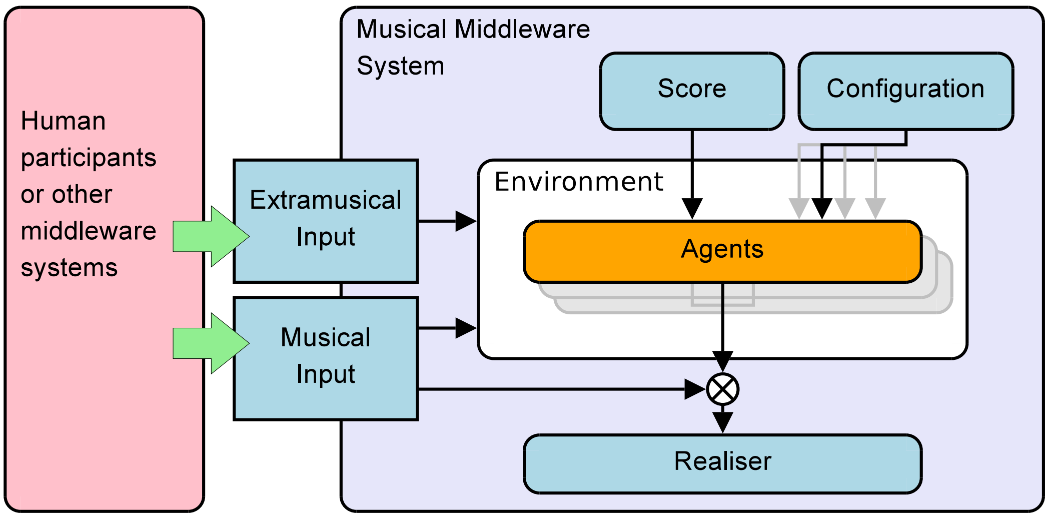


Figure 4.2: Agent System Overview

level than playing notes. This includes being able to take an "intelligent" approach to creating music, balancing low level and high level issues in order to create music which fits the requests of the humans, and "works" musically.

4.2 Agent System Model

An overview of the agent system can be seen in Figure 4.2. A breakdown of the components, and where their full treatments may be found is:

Agents are musical agents. They communicate by passing messages, both musical and non musical. The design of agents is discussed in Section 4.2.5.

Score is some high level representation of a piece of music for the agents to play. It is discussed in Section 4.3.7.

Configuration provides specification to the agent system as to what agents should be run, and any other necessary parameters. It is generally implementation specific, and hence discussed in Chapter 7.

Realiser transforms music from the representations used by agents into sound or some useful format. It is discussed in Section 4.2.2.

Environment provides an environment for the agents to exist in which fits the specifications from the system model in Section 6.2.2 and performs any other physical simulation which is desired. This is discussed in Section 4.2.1.

4.2.1 Environment

In most common musical settings, each performer will hear the output of each other performer, and they will hear it at approximately the same time - this is a natural effect of the transmission of sound. In an agent system, agents only receive messages which they have been specifically sent. Section 6.2.2 set out a model for the production and perception of music, designed with networked agents in mind. In this model, time is treated as happening in discrete blocks of time. These blocks are referred to as *timeslices*, with a *fragment* denoting the music which happens in a specific *timeslice*. This is a match for the message passing in agent systems — a message can represent a single fragment of music.

The rest of this section deals with the questions

- how are the agents to send messages to each other?
- how are the temporal aspects of passing musical messages around managed?

4.2.2 Realiser

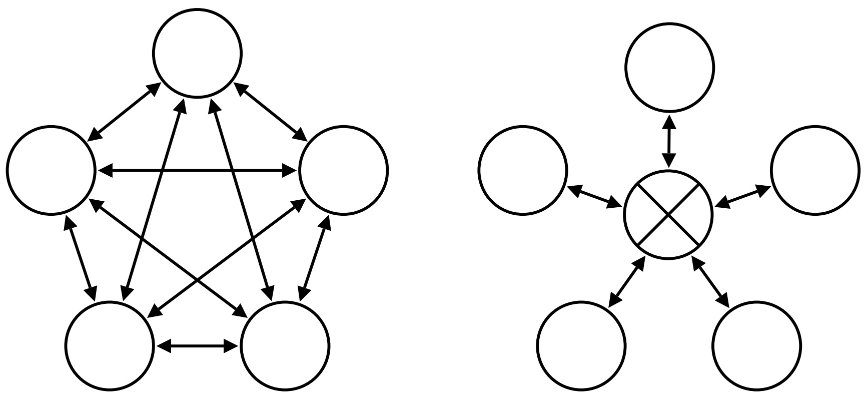
While not strictly part of the environment, the realiser is a bridge between the agent's environment and the real world, and so is discussed in this context. The realiser's job is to transform representations internal to the agent system into music in a useful form for the rest of the world. In general, this could be many different things; here, it will be assumed that it is some device which produces audio in real time.

The assumptions made about the realiser from a system design perspective are:

- it operates in real time.
- it accepts fragments of music
- there is a certain minimum time between passing a fragment to the device and that fragment starting to play.

4.2.3 System organisation

In the simplified model of music production and perception (Section 6.2.2), each agent can hear perfectly the output of every other agent; hence, a mechanism is needed for sending the output of each agent to each other agent. In order to do this, a protocol must be designed which details what messages are sent, between which parties and how



(a) Decentralised (b) Centralised

Figure 4.3: Configurations for Musical Multi-Agent System

the message flow is controlled. For this discussion, only messages which represent a fragment of music are considered — extramusical communication is ignored.

Firstly, there is the question of what topology is used to send messages between agents; there are two simple choices — see Figure 4.3:

Decentralised Every agent sends messages to every other agent.

Centralised There is a central server agent, which all the other musical agents send their output to. This agent then collates everybody's output, and disseminates it.

The next question is how the flow of messages is to be controlled. Two possibilities are:

- each agent has its own sense of time, and sends the next fragment of output when ready.
- a central agent triggers the sending of messages, by requesting output from all the other agents

The proposed system system uses the latter technique, with a centralised server, which controls the timing of messages and acts as a central message distribution server for the following reasons:

- fewer messages need to be passed. For each timeslice, every agent sends a single message with its output, and receives one message with the collated output of the whole system. This is $O(n)$ messages, rather than $O(n^2)$ for the fully connected system. The total amount of information transmitted increases marginally, but

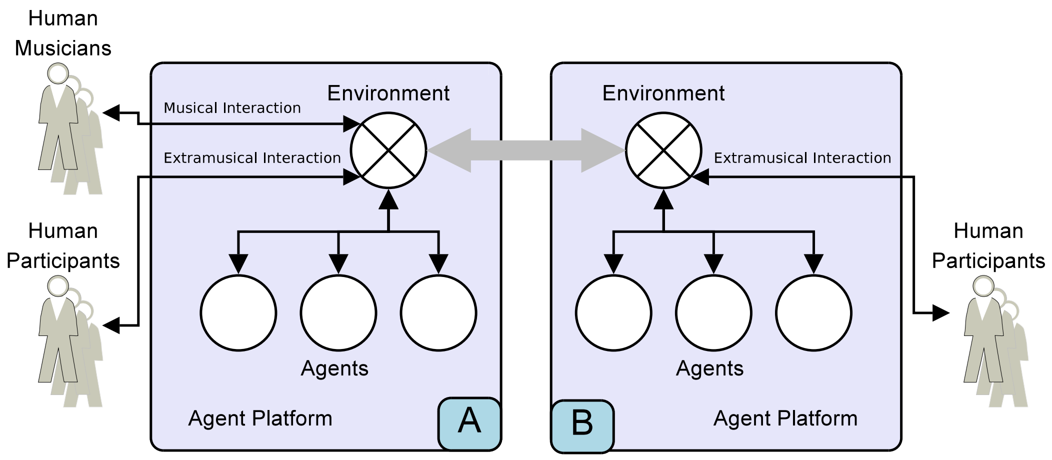


Figure 4.4: Two agent platforms and human participants collaborating

for many agent systems the time and resources involved in sending messages often outweighs the cost of sending the data which is in the message.

- If another agent system is involved, then it need only communicate server to server - (see Figure 4.4). Here the server on platform A receives output from all the agents on platform A, from the human performer, and the collated output of platform B. It then sends all of this output to the agents on platform A, the server of platform B, and outputs sound for the human participants in the vicinity of A.
- by centralising the timekeeping, there is only one part of the system with strict realtime requirements — the central server is required to keep the realiser supplied with output. All of the rest of the operations in the system must take place in a timely fashion, but do not have the same strictures placed upon them. This greatly simplifies the task of writing individual agents, as their contract with the rest of the system can be made very simple.

The system hence has a central server whose responsibilities are:

- Requesting output from all the agents involved.
- Collating their output (along with any external input), and disseminating it.
- Sending the output to the realiser at the correct times.

This is to some extent against the spirit of *agency*[1], as it creates a bottleneck and a

[1]See Section 2.4 for a discussion of the spirit of agency.

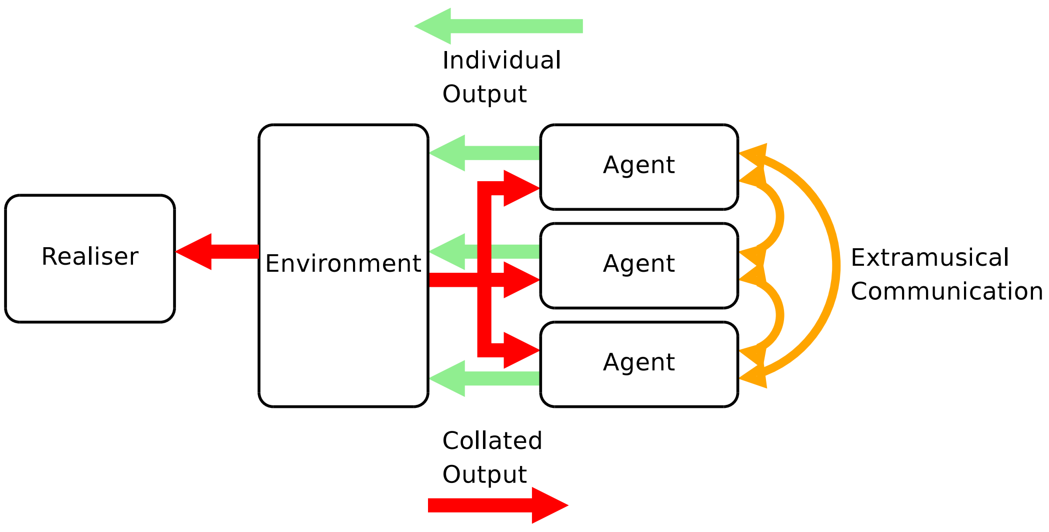


Figure 4.5: Overview of Environment functions

single point of failure. However, the operation of this server should be considered as more analogous to providing the environment in which the agents exist.

The server deals only with distributing the music produced by the agents and realising it as sound (or MIDI), and does not deal with any extramusical communication — agents must do this for themselves. The responsibilities and organisation of the server can be seen in Figure 4.5.

This server will be referred to as the `Environment` as it provides an approximation of the environmental properties which would occur when several humans were sharing a physical space.

4.2.4 Temporal aspects

It is the responsibility of the `Environment` to ensure that music is processed in a timely fashion; timing is one of the primary qualities of music, and hence one to which a lot of care must be paid.

The `Environment` has the following tasks to perform:

- request musical output from all of the agents who are playing.
- disseminate collated output to all agents who are interested.
- send the collated output to the realiser for playback.

There is a set of *timing goals* which should be maintained to ensure correct operation of the system, each with different penalties for failure:

Time	Environment	Musician
t_k	Fragment k starts playing	
$t_k + t_{lat} - t_{sched}$		Receive fragment k
$t_{k+1} - 2t_{lat} - t_{sched}$	Request fragment $k+1$	
$t_{k+1} - t_{lat} - t_{sched}$		Receive request for fragment $k+1$
$t_{k+1} - t_{sched}$	Schedule fragment $k+1$ to play and send it to musicians	
t_{k+1}	Fragment $k+1$ starts playing	

Table 4.1: Network Timing Table

$TG_{realise}$: each fragment of music must begin playing when the previous fragment finishes; failure here will result in gaps in the sound, or inaccurate timing depending on the playback method used.

$TG_{responses}$: each agent must have time to respond with its output before that fragment begins playing; failure here will mean that the output of one or more agents drops out for a fragment.

$TG_{processing}$: the earlier agents receive the output of other agents, the more processing can be performed to construct responses; this is not a hard goal, more a parameter to be maximised.

There are three variables governing this:

t_{sched} The time it takes for the realiser to schedule a fragment to be played. The `Environment` must have given a fragment to the `Realiser` at least t_{sched} *before* the start of the timeslice in which the fragment should be played in order to ensure that the fragment is played correctly.

t_{lat} The message latency - the time it takes from one agent sending a message to the message being received. In this analysis, this is approximated as being constant.

t_{frag} The length of a timeslice (equivalent to the length of time it takes to play a fragment).

The assumption is made that network tasks take an order of magnitude more time than processing tasks, so processing tasks are ignored here. A contract is placed on

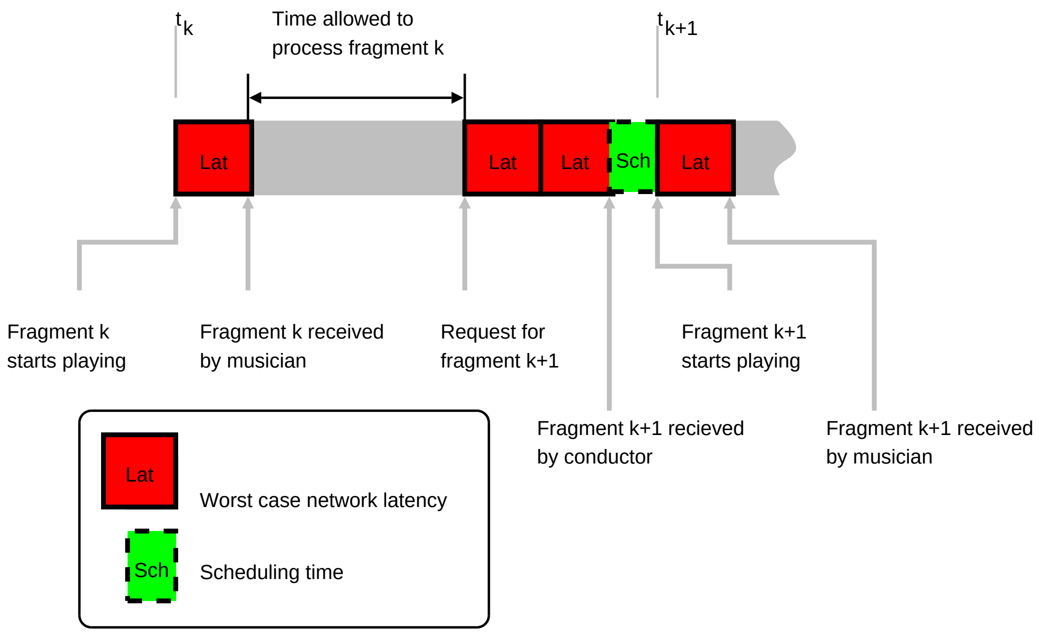

t_k
Time allowed to process fragment k
t_k+1
Lat
Lat
Lat
Sch
Lat
Fragment k starts playing
Fragment k received by musician
Request for fragment k+1
Fragment k+1 starts playing
Fragment k+1 recieved by conductor
Fragment k+1 received by musician
Lat
Worst case network latency
Sch
Scheduling time

Figure 4.6: Agent system timing diagram

agents such that they must always reply *immediately* with their next fragment when asked, so this is assumed to be instantaneous. Table 4.1 and Figure 4.6 describe the timing sequence involved in playing a fragment — fragment k, which covers the timeslice from t_k to t_{k+1} — and the surrounding timeslices. This describes the tightest possible bounds on all of the timings which avoids conflict with $TG_{realise}$ or $TG_{responses}$.

In a more narrative form, the sequence is:

- Starting with $TG_{realise}$, in order for fragment k to be playing at time t_k, it must have been sent to the output device at time $t_k - t_{sched}$. Assuming that this is the first time the `Environment` has the collated output of all the agents, it will disseminate fragment k to all the agents at this time.
- At t_k, fragment k begins playing.
- At $t_k - t_{sched} + t_{lat}$, the agents receive each other's output for fragment k.
- the fragment continues playing, and the agents work on constructing their next responses.
- the next event is the `Environment` asking for fragment $k+1$; in order to play fragment $k+1$ at time t_{k+1}, there must be time for:

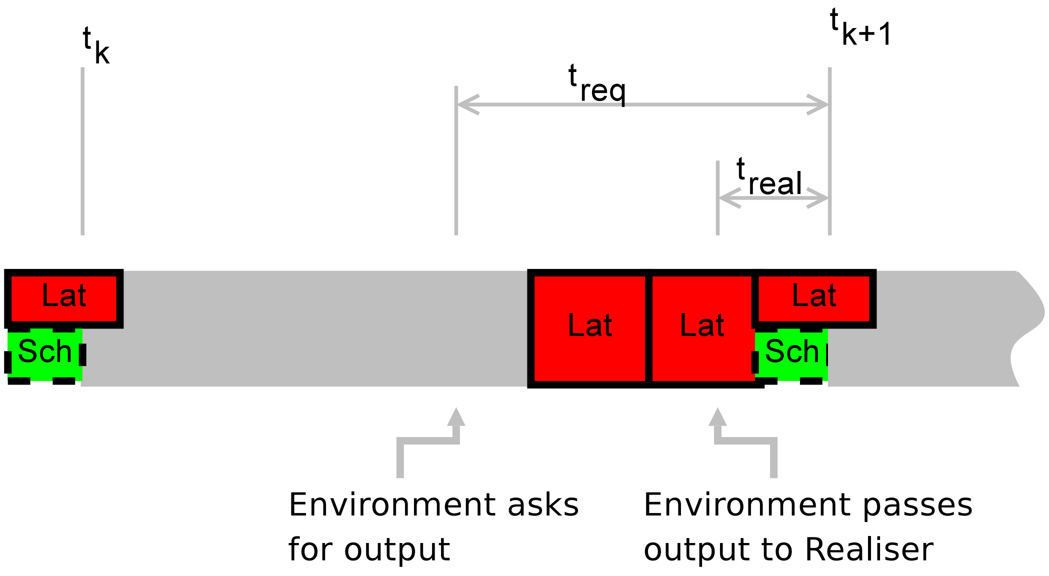

t_k
t_{k+1}
t_{req}
t_{real}
Lat
Sch
Lat
Lat
Lat
Sch
Environment asks for output
Environment passes output to Realiser

Figure 4.7: Timing points for the Environment to determine

- the environment to request the next fragment from the agents,
- the agents to respond,
- fragment $k+1$ to be scheduled;

hence this occurs at $t_{k+1} - 2t_{lat} - t_{sched}$.

- the agents receive this request at $t_{k+1} - t_{lat} - t_{sched}$, and the `Environment` receives their responses at $t_{k+1} - t_{sched}$, just in time to schedule it for playback.

In a less theoretical situation, it is likely that t_{lat} and t_{sched} are not constant, so an amount of leeway must be added. It is hence up to the `Environment` to determine two timings — t_{req} for requesting output from the agents, and t_{real} for sending all the output which has been received (for this timeslice) to the `Realiser` (see Figure 4.7). The details of this are implementation specific, but should be guided by the analysis here.

Looking at optimisation of the system, the main characteristic of interest is how quickly agents can respond to new events. There are two types of events which must be taken into account: those from other agents, and those from external processes (such as human musicians). Figure 4.8 shows the order of events for the rest of the agents to respond to an event produced by agent A. Since the agents have access to each other's output before it occurs in realtime, the time delay is minimised. If t_{IR} is the response

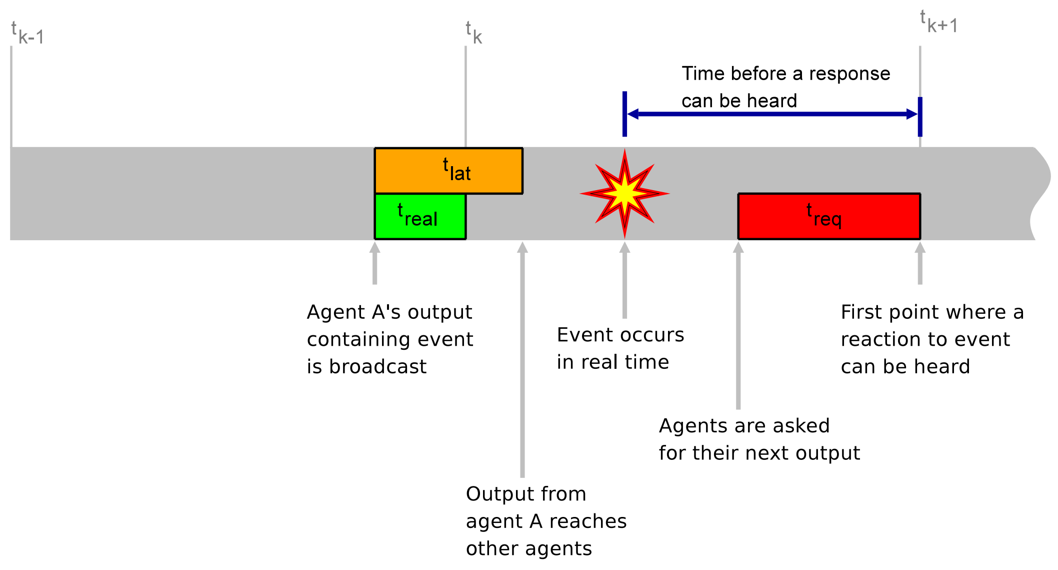

t_{k-1}
t_k
t_{k+1}
Time before a response can be heard
t_{lat}
t_{real}
t_{req}
Agent A's output containing event is broadcast
Event occurs in real time
First point where a reaction to event can be heard
Agents are asked for their next output
Output from agent A reaches other agents

Figure 4.8: Timing diagram for reactions to internal events

time for internally generated events, the bounds are $0 \leq t_{IR} \leq t_{frag}$ relative to the time the event is heard. Assuming for now that realtime musical events are disseminated to the agents under a similar protocol to the internal agent's events, except for the caveat that events are not available until they have happened, and hence can only be disseminated in the next timeslice, Figure 4.9 shows the flow of events. If t_{ER} is the response time for externally generated events, the bounds on responsiveness are now $t_{frag} + t_{req} \leq t_{ER} \leq 2t_{frag} + t_{req}$.

In most cases, $t_{frag} > t_{req}$, so the main limiting factor in both these inequalities is t_{frag}, the fragment size. Hence increased responsiveness can be gained by reducing this; unfortunately, there are costs:

- the smaller the fragments are, the more fragments must be passed around, and the more the message passing overhead becomes a burden.

- smaller fragments necessarily impose tighter timing constraints on the system, resulting in a higher chance of missing deadlines.

The choice of t_{frag} (just as t_{req} and t_{sched}) is an implementation decision. It is entirely conceivable that it could be altered dynamically by the system to adapt to changing network conditions; it may even be possible to run different agents at different fragment sizes.

If a situation is reached where $t_{frag} < t_{req}$, then agents will be asked for their next

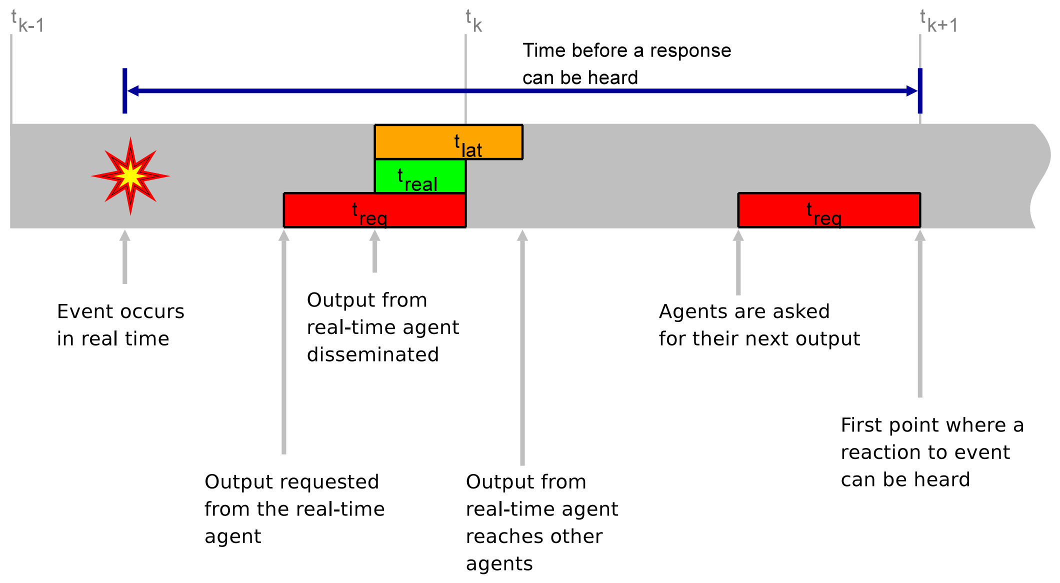

t_k-1
t_k
t_k+1
Time before a response can be heard
t_lat
t_real
t_req
t_req
Event occurs in real time
Output from real-time agent disseminated
Agents are asked for their next output
First point where a reaction to event can be heard
Output requested from the real-time agent
Output from real-time agent reaches other agents

Figure 4.9: Timing diagram for reactions to external events

fragment before receiving the previous one — they will be playing a fragment behind, and hence $t_{response}$ will be increased by t_{frag}. This means that if network conditions deteriorate, the music will not stop, it just becomes less responsive to input; effectively, the better the network and the quicker the implementation, the faster agents can respond, and the more information they have to work with.

There is a less fortunate implication of the structure shown in Figure 4.9: events passing through the Environment cannot be scheduled by the `Realiser` in real time. The consequences of this are:

- if the events need to be played by the `Realiser` — for example if they are MIDI events to be sent to a synthesiser — an alternative route must be found for them.

- it is not possible for two communicating agent systems in different locations to work with musical input from humans (assuming that there is a significant delay in communication between the two systems):

  – in a single agent system, all the agents are producing their output one fragment before it is heard — ahead of real-time.

  – A single agent system works with human input because the agents are working ahead of real time, and processing input from the humans after it happens

- if two systems composed only of agents are connected, then the agents can continue to work ahead of real time; because the agents are sending their output before it is heard, the time lag between the two systems can be dealt with.
- adding human input to one of these systems is no problem, as all the agents can continue to work ahead of time, reacting to the human input after it happens
- having humans on both systems causes a problem; humans cannot provide their output before it happens, so humans on one system can only hear the output of humans on the other system after a time delay, which is not a generally accepted mode of performance (see Chew et al. [2005] and related work for more discussion in this area).

4.2.5 Specification

To sum up the decisions made in this section, a concise specification can be given for the behaviour of the environment. The Environment is responsible for:

- turning representations of music into sound.
- asking for musical output from all agents in a timely manner.
- disseminating the output of each agent to every other agent.

Musical agents have only one concrete responsibility: to respond with their next fragment of music immediately. This is summed up in Figure 4.10.

4.3 Musical Agent Architecture

This section presents an internal architecture for a musical agent, in response to the capabilities and functionality discussed in the previous section. The architecture is blocked out, and then each component is detailed, along with how it relates to the proposed specification.

In talking about agent architecture, there are two useful overviews: firstly, because an agent does not operate in isolation, Figure 4.2 shows the configuration of the entire

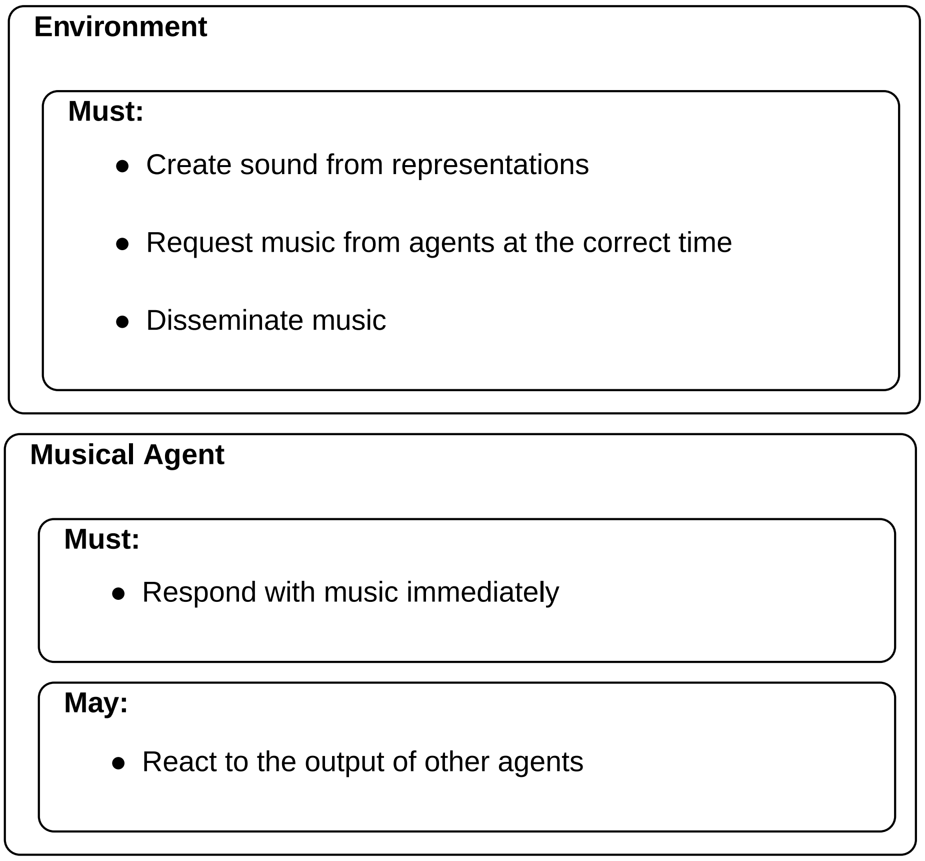

Environment
Must:
● Create sound from representations
● Request music from agents at the correct time
● Disseminate music
Musical Agent
Must:
● Respond with music immediately
May:
● React to the output of other agents

Figure 4.10: Contract between the agents and the environment

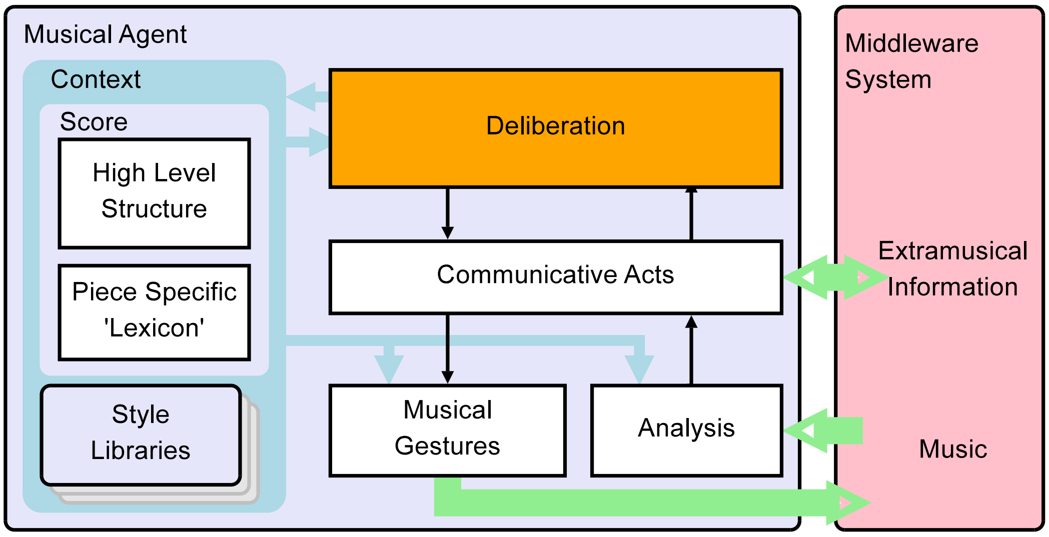


Figure 4.11: Single Agent Overview

agent system; secondly, Figure 4.11 shows the anatomy of a single musical agent.

4.3.1 Inputs and Outputs

The system takes two kinds of input - atemporal and temporal. There are two atemporal inputs; firstly, the score for the piece of music. This has been created by a composer, and gives specific details of the piece at hand (this is discussed in Section 4.3.7). Secondly, the user may configure their middleware system in some way. This may include choosing which agents are available, loading specific agents which can work with certain types of music, loading libraries to deal with musical styles and loading personal preferences, techniques and knowledge.

The temporal inputs are receiving music from the middleware system (both from other agents and from humans), sending music back to the system for dissemination and realisation, and any extramusical communication which is carried out as the piece progresses.

4.3.2 Context

Each agent maintains a context of the piece which is being played, in order to create music and interpret the music of others. Some of this context is specific to the piece at hand, and is termed the score. The score contains a high level structure for the piece, detailing how sections of music fit together, and what happens in each section. It may also contain a “Lexicon” for the piece - a repository of musical material, some of which can be explicitly labelled. This material is available to the agents to use as necessary, and also as a guide in interpreting the actions of others. Certain fragments may be given special significance, such as the change rhythms used in African and samba drumming.

The style libraries are similar to the Lexicon for a piece - they contain material related to that musical style. This can include common chord progressions, rhythms, ornamentation etc. This models the knowledge which would be expected of a practising musician in a certain genre.

4.3.3 Deliberation

Deliberation is central to both the autonomy of agents in the system, and the perception of them as equal partners in an improvisation. The deliberative system is responsible for creating a high level plan of what the agent is to do. The deliberative system takes as input the musical context and the communicative acts performed by others, and outputs a series of communicative acts to enact and modifications to the musical context. This will be more fully discussed in Chapter 9, but in brief, three types of work can be performed: planning, maintenance and feedback.

In planning, the agent can:

- interpret the score, to determine what sections to play, and what bearing the score has on the agents actions - e.g. whether the agent should adopt a certain role at this point in the piece.
- manage its intentions - dealing with conflicting interests, creating new intentions based on the score.
- decide how to react to the input from the other participants; whether to passively accept their input, to expand on it, to reject it.

The result of this work is a series of communicative acts which the agent will enact. There is also a connection to the maintenance component, concerning how the actions of others prompt the agent to modify its own context.

Maintenance involves altering the musical context based on the actions of other agents. This may be as simple as changing what section the agent is playing, or it may involve changes to the given score. This means that we require a representation of music compatible with inference. This has some characteristics which are addressed below (Section 4.3.7).

Finally, the agent may have a mechanism for integrating feedback, both within a single interaction, or over the course of several interactions.

4.3.4 Communicative Acts

The need for a level of communicative acts in the system has been discussed; this is one of the distinguishing features of middleware, and is part of the concept of agent system.

The communicative act layer takes as input sequences of communicative acts from the deliberative layer. It also receives the actions of other agents, both from the analysis of their musical behaviour, and through a formal channel from the rest of the middleware system. The output of the section is the sequence of communicative acts in a suitable format for input to the generation system.

In order to function effectively as middleware, it is desirable to allow direct access to this communicative act layer. This enables a variety of extramusical interaction, some examples being:

- Other agents in the system can give explicit information about what they are doing, providing a cheap and accurate view onto mental state
- High level interfaces can work with the system at this level, giving a composer/performer a way to influence the system in realtime, by suggesting musical and altering sections of the score.
- With suitable conversion, other modalities may be mapped into the musical space providing a means for the system to react to the actions of physical performer, or participants in an installation.

4.3.5 Analysis

The analysis layer takes as input the musical output of the other participants, and the current musical context. By comparing the actions of others to the context, it extracts a set of low level features which are part of the musical context, and a sequence of communicative acts which are passed up to higher levels.

4.3.6 Generation

The generation section takes as input the entire current context, and a series of communicative acts. The communicative acts can be rendered to music in the given context, which is then output to the rest of the system.

The sequence of communicative acts to be expressed may only define a small section of the musical surface, and there may well be times when an agent is not executing any acts at all. As an example, when supporting a soloist, one's playing would be relatively free of new ideas in order that the soloist's voice is more clearly heard (much as one is silent in conversation to allow another party to speak). This means that musical acts are a supplement to the generation process, and the bulk of the musical surface may have to be generated from the score and the current context. Material may be extracted from libraries, altered to fit, and used; the score may define certain aspects of the surface which should be adhered to; the lexicon built up for the current piece may provide data which can be adjusted and used. In these ways, the generation system attempts to create musically correct output, modulated by the communicative actions the agent is attempting.

4.3.7 Specification

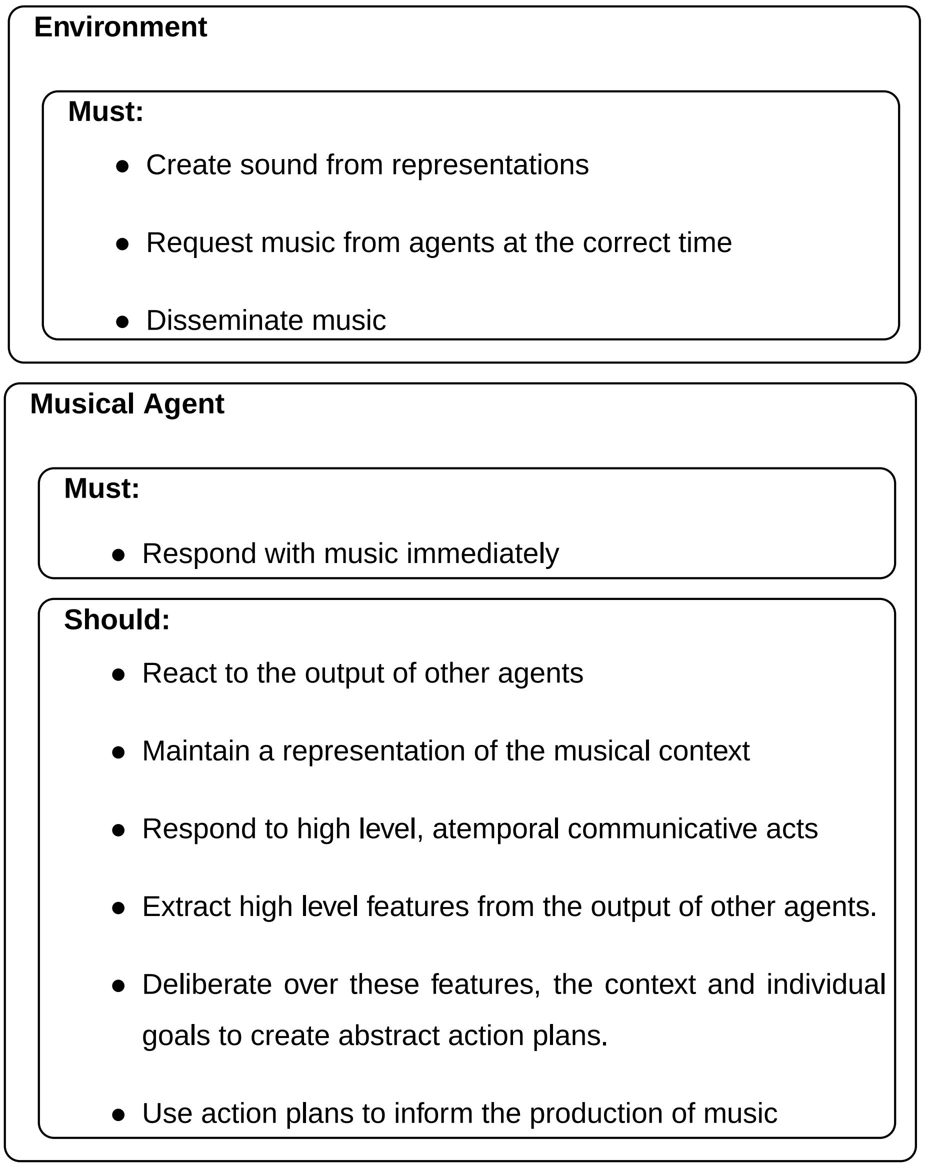
In light of this discussion of Musical Agent capabilities, the specification in Figure 4.10 can be expanded to that shown in Figure 4.12. The capabilities outlined in this section have been labelled as "should" — it is possible to make a musical agent which does not perform these functions, but it is not in the spirit of musical agency. There may be valid reasons for creating agents with alternative functionality, though, particularly for interfacing with the real world, e.g. creating representations of humans in the agent system. Reacting to the output of other agents has been upgraded from "may" to "should", as this is more necessary in the context of intelligent musical agents than it was in the context of an architecture for managing the low-level transfer of music in Section 4.1.2.

4.4 Music Representation

This section deals with the specification of a representation language for musical agents, which will be called MRA (Musical Representation for Agents). Some key components of the language are identified, but the specifics of the language are not detailed until Section 7.2.4

The music representation system has two different types of music to deal with — music-as-scored, and music-as-played. In this architecture, these are represented using the same objects. This is partly a way to simplify the system design, but more importantly it helps to keep in mind the idea that in this system, there is not a distinct line between music-as-played, and music-as-scored: firstly, scores may have varying levels of specificity, which get filled in by different parts of the system, and secondly a representation at one level is almost always passed to a lower level at some stage — even the music-as-played is later passed to a synthesiser which determines exactly how the music sounds. The main issue arising from this dual use of the representation is that it is not clear from looking at a piece of MRA whether it has been played, or is input for some system *to* play; programatically, however, this is not generally a problem, as each subsystem knows what it is doing with the music it receives.

There are three forms which music representations are present in, each of which



Environment

Must:

- Create sound from representations
- Request music from agents at the correct time
- Disseminate music

Musical Agent

Must:

- Respond with music immediately

Should:

- React to the output of other agents
- Maintain a representation of the musical context
- Respond to high level, atemporal communicative acts
- Extract high level features from the output of other agents.
- Deliberate over these features, the context and individual goals to create abstract action plans.
- Use action plans to inform the production of music

Figure 4.12: Complete contract for musical agent system

have different design goals:

- in the agent's memory, as some form of "musical objects",
- on the network,
- in files for creation and storage.

This section does not deal with the specifics of how music is represented in these different situations; it looks at the general features which the different representations must support.

4.4.1 Design Criteria

In Chapter 4, some capabilities for the system as a whole were set out. Working from these, and adding points as necessary, a set of design criteria for a musical agent language are derived:

Structured There must be enough structure to support reasoning; this should be compatible with the musical structure to be represented.

Addressable Agents must be able to address specific sections of the representation in order to communicate about them.

Flexible It must be possible to create scores which are not entirely linear; this includes changing the number of times a section of music repeats, and changing the ordering of sections within a piece.

Non-specific The representation should be applicable to as many different forms of music as possible.

Open The language should be platform independent and preferably human readable (both for composition and easy debugging). It should be possible for completely different agent systems to play music together.

Powerful It should be possible to represent everything we need about a piece of music.

Extensible The representation should be extensible to cover new types of music, and new features of existing music.

Natural If the representation can mimic the way in which music is currently written, there is less of a shift for musicians starting to use the system.

Piece

Name: (<string>)

Tempo: (<string>)

TimeSignature: (<integer>)/(<integer>)

Section(name=”main”)

Section*

Section

Name: (<string>)

Derives: (<section-name>)?

Ordering: (”sequential”|”dynamic”|. . .)?

NumRepeats: (<integer>)?

RepeatUntil: (<repeats specifier>)?

Either:	Section+
Or:	Length=(<float>)
	Channel*

Channel

Name: (<string>)

(TemporalEvent)*

TemporalEvent

Onset: (<float>)

Note extends TemporalEvent

Pitch: (<pitch-specifier>)

Duration: (<float>)

Velocity: (<float>)?

Items ending with a ’:’ indicate attributes which may take values; others indicate that objects of that type should be present. Items with no modifier are mandatory, ’*’ indicates any number, ’+’ indicates one or more and ’?’ indicates zero or more occurrences.

Figure 4.13: Overview of music representation specification

These capabilities are used as guiding principles in the design of the music representation system.

4.4.2 Musical Structures

The fundamental tenet of the MRA system is that the representation of a piece can be broken down into two types of object; structural objects define the order and structure of the parts of a piece, and objects attached to this structure define what happens at various points in the piece. A further specification is that structural elements are arranged in a hierarchy, and that all structural elements are named. This is designed to address the *structured* and *addressable* design criteria. The fact that a wide range of objects can be attached to these structural units is designed to address the *non-specific, powerful* and *extensible* criteria. The criterion of *flexibility* is addressed by the way in which sections relate to each other, and the *open* and *natural* criteria are implementation issues, and covered in Section 7.2.4.

Figure 4.13 gives an overview of the specification developed here.

4.4.2.1 Section

Sections are the basic building blocks of musical structure in MRA. At one end of the scale, a section is designed to equate to the section markings given in many musical scores, representing something of the order of a few bars of music; these sections may be composited into larger sections, relating to concepts such as verse, chorus etc. or larger scale units like movements. A special case of section is then used to represent an entire piece of music.

Each section has a name, which must be unique. The hierarchical nature of sections means that a section somewhere down the tree can be uniquely addressed by creating a path of the names of the sections above it.

Sections may be either branches or leaves:

Branches contain other sections, and as a result cannot contain any temporal events (Spans and Channels) and may not have a length set.

Leaves do not contain other Sections, but may contain Spans and Channels, and should have a length set.

To avoid writing out similar attributes repeatedly, Sections inherit attributes in two separate ways: firstly, a Section will inherit attributes from its parent in the hierarchy;

secondly, a Section may specify another Section which it inherits attributes from. This allows for the creation of Sections outside of the main tree, which act as base Sections for the real Section. In the case of attributes which are specified in multiple places, specifications in a Section take priority, then values from explicitly specified Section, with hierarchically inherited values being weakest.

Sections support the following features:

Hierarchy A piece consists of a main Section, which is the root of a tree that defines the entire piece. Playing of a piece consists of playing the Section marked `main`. It is possible to create Sections outside of the `main` Section: these can be used as necessary throughout the piece, but are not part of the usual execution order.

Ordering Since Sections define the structure of the piece, there must be a way to control their ordering. There are several mechanisms for this:

- **Sequential** If nothing else is specified, Sections run through in the order they are listed in the score. When a Section finishes, the next Section in its parent will play. If it is the last Section in its parent, control is passed to its parents successor (the next Section in the grandparent). Since only leaf sections are played, this is equivalent to producing an ordered list of the fringe.
- **Repetition** A Section may specify that it should be repeated several times. This can be a number, or it can repeat indefinitely until some external event triggers moving to the next Section.
- **Specific** A Section may specify which Section should be played next by giving a path to the appropriate Section.
- **Dynamic** A Section may specify a decision process over its children to decide which Section to play next. In the current implementation, the sequential ordering is an example of a process over the children of a Section. This can easily be changed to allow stochastic orderings.

Attributes Sections use attributes to define characteristics of the piece. An attribute is a name, with a value. The value can be arbitrarily complex, depending on which particular attribute it is. If an attribute is constant over the course of a Section, it is atemporal - examples might be tonality, meter, the rhythm to use etc. These attributes are available to all Sections, and are defined directly in the Section.

Time dependant attributes - such as notes, accents or chord sequences - must be defined in “Channels”, and are only available to leaf Sections.

4.4.2.2 Channels and Temporal Events

In order to represent events which happen at a certain time within a section of music, channels and temporal events are used. A temporal event is some kind of object, with associated time information, while a channel is an ordered collection of similar types of temporal events. This is intentionally left open, as different styles of music will demand different representations. The possibilities include dynamic curves, patterns of accents, chord sequences, lyrics and so on — essentially, anything which would be found in a traditional score, indicated at certain places on the metric grid.

However, one particular specialisation is demanded by the architecture: a fragment is a channel which represents a small portion of the musical surface[2] as music-as-played, which can be exchanged over the network. In much of the following work, this translates to being filled with notes, which are relations between two temporal attributes (onset and duration) and other properties such as pitch and volume.

4.5 Conclusions

This chapter has:

- presented motivations for musical middleware, and target capabilities.
- developed an implementable model of how this system functions.
- set out specifications for how the components interact.
- defined a language which is used to represent music in this system.

[2]this is related to the discussion of the temporal mechanics of exchanging music in Chapter 6

Chapter 5

Musical Acts

This chapter presents a top-down view of musical acts, which provides the motivation for the subsequent development of a formal model of musical actions

5.1 An Intuitive Notion of Musical Acts

The use of communicative acts in multi-agent systems is widespread — see for example [FIPA] or [Wooldridge, 2002] — as defining formal semantics for communicative actions allows the modelling of patterns of interaction as protocols based on the exchange of communicative actions, and allows agents to reason about the beliefs and intentions of other agents (e.g. Herzig and Longin [2002]). This chapter takes a top-down approach to exploring what an analogous formulation for musical interactions looks like.

This analogy is suggested by the musical literature which refers to musical improvisation and performance in terms of "conversation" or "discussion" between the musicians [Coventry and Blackwell, 1994, Walker, 1997].

There are several structural differences between musical interaction and the exchange of communicative actions typical of multi-agent systems:

- musicians typically play simultaneously with other musicians, rather than alternating discrete actions — however, there is some support for the idea that even though the musicians are *playing* at the same time, they may not all be *communicating* at the same time [Pelz-Sherman, 1998].

- it is generally necessary to model interaction between more than two agents, which is the case usually treated in dialogue — with some exceptions (Atkinson et al. [2005], Walton [2004]).

- no use is made of the notion of “literal meaning” of a musical statement that might correspond to the literal meaning of a natural language statement. It is the polysemic nature of music [Cross, 2003] which makes it more suited to this pragmatic mode of enquiry.

Murray-Rust and Smaill [2005], Murray-Rust et al. [2006] are previous attempts to formalise Musical Acts in the context of a multi-agent system.

5.1.1 Characterisation of Musical Acts

In order to progress, a more detailed picture must be built up of what exactly a musical act is, and what they are useful for. The rationale for developing this theory is to create a narrative “plan-view” of a musical interaction, which sheds light on the actions musicians take and the reasons for those actions. Some qualities which these acts must have are:

Embodiment through the production of music; much as speech acts are embodied through the production of utterances within certain contexts, 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 — it should be an attempt to change the state of the world or the actions of others by its production, in a more significant sense than the mere fact that it has been produced.

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. So, the act must be conceptualised within the context of a certain musical situation, with a certain expectation of understanding from the other musicians.

So given these requirements, some questions are raised about how musical acts exist, and how they may be determined:

What kinds of changes in the world are expected? How can acts be detected? How is the intention behind an act determined? Does this relate to ideas such as soloist/accompanist? Can acts occur in the context of a string quartet, or are they only applicable to improvised musics?

When musicians play together, there is generally some set of musical structures which all of the musicians would agree on. For example tempo, chord structure, song structure and so on. In different musical styles, this shared understanding may take different forms: in an orchestra it could be a printed score along with a set of rehearsed directions about how to play it, in a free jazz ensemble, it may be a diffuse, implicit knowledge of what is *currently* happening on several different levels. This shared representation comes from two sources — pre-existing structures, such as scores or musical traditions, and extemporaneously created material which arises as part of the interaction between the musicians.

During the course of the musical interaction, musical output can be roughly divided into two partitions: that which is expected according to the shared representation that the agents have, and that which is unexpected or novel. This decision needs more analysis, as it may be that only certain aspects of the music are predicted, while others are novel — for instance, playing the expected melody with a different rhythm, or unexpected expressive character, playing the expected rhythm, but accenting different notes. Hence, certain features of the music played can be seen as novel, and hence a vehicle for musical change. This means that all of the repetitive and commonplace aspects of the music can be ignored — the drummer keeping time, the fact that the string quartet are playing the right notes — in favour of analysing the novel occurrences, such as the phrasing structure the drummer uses, and the way that the expressive aspects of the violin's playing change in response to the cello.

This casts a musical act as an attempt to alter the shared representation of music which the musicians have, by playing something which differs from the shared representation.

5.1.2 An Example: Little Blue Frog

"Little Blue Frog" [Davis, 2000], on the Columbia/Legacy reissue of "Big Fun" is used as an example for what a Musical Act analysis of a piece of music looks like. The analysis is carried out from a personal listening perspective — that is, without recourse to transcriptions or other theoretical explanations, using only my individual musical competences, and to offer a view of what I personally found to be the most interesting parts of the interaction. This is fully in keeping with the philosophical thrust of Musical Act Theory — that meanings are extracted and used by individuals, who have their own capabilities, idiosyncrasies and deficiencies, and any model must take this into account. Figure B.2 in Appendix B presents this analysis, while a small section of this is shown in Figure 5.1.

Time (s)	Instrument	Performative	Description
2:00 - 2:13	Trumpet	INFORM	A spiky, stabbing phrase, based on scale 2
	Clarinet	CONFIRM	briefly seems to agree with the trumpet.
2:13 - 2:29	Bass clarinet	CONFIRM	Confirms scale 3
	Trumpet	DISCONFIRM	Ignores bass clarinet, and continues with stabs
	Clarinet	DISCONFIRM	Ignores bass clarinet, and continues with lyricism in scale 2
2:29 - 2:43	Trumpet Clarinet Bass Clarinet	ARGUE	All play lyrically, with clarinet on scale 2, trumpet on 1 and Bass Clarinet in 3
2:43 - 3:08	Trumpet	PROPOSE	proposes a resolution, by playing stabs which fit with any of the scales
3:03 - 3:08	E-Piano Vibes	CONFIRM	supports the trumpet's resolution

Table 5.1: Example Musical Act analysis - Little Blue Frog (Miles Davis)

It should also be noted that this analysis was carried out from the point of view of a listener who was not party to rehearsals or discussions prior to the performance of the piece; this means that initially, my representation of what would occur in the piece was empty, or contained general stylistic expectations of what I think a Miles Davis fusion piece is likely to sound like. To the musicians who were playing, a different set of musical features would be seen as novel and intentional, and a lot of the changes which I notice would be expected due to whatever score had been agreed on beforehand.

5.1.3 Performative Acts

The performative labels attached to the musical act analysis presented above were constructed in an ad-hoc manner, so now the terms which were used should be defined and discussed more thoroughly. However, this is not a formal or exclusive definition — more a sketch of the type of *story* which musical acts are aimed at telling.

Propose occurs when an agent introduces a new musical idea; this should be an idea which does not conflict with any of the material which is already present — for example, introducing a harmonic structure when previously only percussive strikes were being played, or introducing a melody when previously only chords and rhythms were being played. In the analysis, the trumpet introduces a lyrical phrase based around a particular scale (0:34-0:55). The performative intention is that the idea being introduced becomes a part of the shared representation.

Confirm occurs when an agent (A) proposes an idea, and another agent (B) indicates amenability to working with this idea; a typical way to carry this out would be for (A) to adopt the idea in its own playing. Again at (0:34-0:55), the clarinet takes up the musical idea introduced by the trumpet. The performative intention is that the new idea becomes part of the shared representation for the agents.

Reject is used by B to indicate unwillingness to adopt A's suggestion — for example, by playing something different to A's idea, or emphatically not taking it up. The performative intention here is that the new idea is not taken up, and does not become part of the shared representation. (1:08-1:35) shows the clarinet suggesting an idea, and the trumpet rejecting it by returning to previous ideas.

Extend happens when an agent presents an elaboration of an already existing idea — for example adding extensions to a chord, playing an elaboration of melody. The intention again is the addition of the new idea to the agents' shared representation. At the beginning of the analysis (0:35-0:55), the xylophone uses the scale which is present, but adds additional, dissonant elements.

Alter is used to change an existing musical idea in some way which has a relationship to the existing material, but cannot be seen as an elaboration of it — for instance changing from a bossa-nova to a son rhythm, or substituting a chord in a chord sequence with one which is harmonically related. (This is a suggested act, not present in this particular analysis).

Request has the intention of causing an action which is not musically related to the musical idea embodied in the action — in the analysis (3:35-4:11), the snare drum plays a crescendo which both provides a point of organisation and a steadily increasing tension which indicates a desire for a new musical direction, without specifying what that direction is.

Argue is a composite act; it happens when several musicians are presenting conflicting ideas at the same time — for example, at (2:29-2:43) where the melody instruments are all playing with different scales.

5.2 Discussion and Future Work

Throughout this chapter, it has been stressed that this is a sketch of what Musical Acts should look like, rather than an exhaustive or accurate definition of "real" Musical Acts which can be used to analyse human playing. It is therefore necessary to give a brief overview of what would be necessary to make this into a usable theory:

- the alphabet or alphabets of actions should be clearly defined. This could be done either by creating a formal semantics for the acts, and exploring the possibilities offered by the system, or by performing a phenomenological study of musical interactions. Murray-Rust and Smaill [2005] is an example of the first approach, which results in five well defined Musical Acts. However, since this theory is intended to explain real-world musical situations, it would be preferable to carry out a study of musical interactions. This would involve recording musical interactions and then attempting to extract and codify acts within these by computational means, performer introspection/analysis and non-performer analysis. Examples of methods which could be used are:

  - performers listen back to recordings of their interactions, and are asked to introspect about their intentions at different points. This eventually leads a set of controlled vocabularies which can be used as the basis for a taxonomy of musical acts.
  - recordings of improvisations are played to listeners, who are asked to indicate where, and between which musicians, communication (and hence intentional actions) takes place.
  - listeners are given a controlled vocabulary to classify recordings which are annotated with points where actions are thought to occur. The agreement between the use of different labels between different listeners can be used as a measure of how much a particular set of performatives captures musical intention.

- once a vocabulary is defined, the semantics of each term in that vocabulary need to be worked out. This should draw on both the natural language meanings of the words used and a formal semantics which allows for a clear description of the effects of the acts.

- In the context of building a musical agent system, the question arises of the computability of these Musical Acts; this is a non-trivial problem, as the *intention* behind an action is not a formal property of the system. In fact, it is common in agent communication languages [FIPA Specification, KQML Spec] to explicitly represent the performative intention of a communicative act. This is an approach which could be used *within* a multiagent system — each agent could label parts of the musical surface with performatives to indicate intentions. However, this would not then extend to an understanding of humans. It would hence be necessary to build a model of intentional behaviour, which could be trained on a large corpus of human data, that was combined with the formal semantics for Musical Acts to give a possible interpretation of musical interactions.

5.3 Conclusions

This section has given an intuitive overview of what musical acts should look like, and an example musical act analysis of an existing piece of music. This provides a motivation for the theoretical work carried out in Chapter 6.

Chapter 6

A Theory of Musical Interaction

6.1 Introduction

In order for musical agents to interact with each other and with humans, it is useful to have a model for how musical interaction occurs which may be computationally implementable. The aim of this chapter is to develop such a model. This is constructed as follows:

- a low-level model of musical exchanges between agents is developed; this represents the "musical surface" — the sound objects emitted by the agents.
- a layer of high level representational features lies above, which the agents extract from the musical surface. These layers are combined to give a model for agents to reason about the musical output and beliefs of others.
- this is used to support a system of musical actions which are extracted from the layer of high level features, and allows the modelling of a musical interaction as a series of related actions.

6.1.1 Different types of actions

When human musicians play together, there are many kinds of action which take place; glances are exchanged, feet are tapped, music is played, particular phrases are played at particular times and so on. For the purposes of this analysis, these will be divided into three kinds of action (see Figure 6.1):

Extramusical Actions occur outside of the musical surface; this includes nods, glances, the foot tapping and body movement used for synchronisation, hand signals for

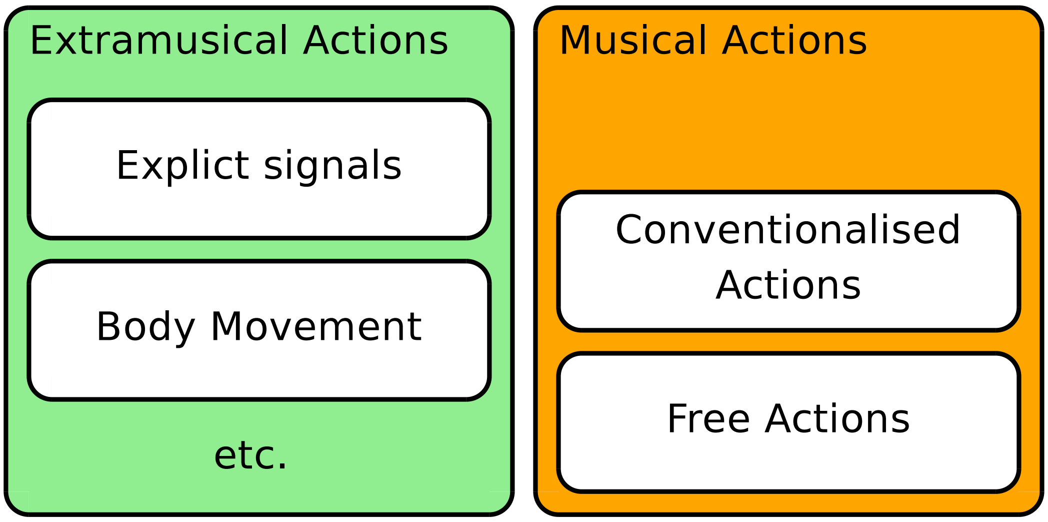


Figure 6.1: Division of actions when playing music

jazz chords, the conductor's baton and every other action which is not contained by the musical surface.

Conventionalised Musical Actions are uses of particular musical phrases with particular conventionalised meanings in certain styles of music at certain points. Some examples of these are:

- the "change rhythm" used in many African drumming traditions, which signals that the next section of the piece should begin.
- whistles used by the *mestre* (band leader/conductor) in samba bands to indicate breaks and new sections.
- James Brown singing "take it to the bridge".
- the *abanico* used in Cuban Son to indicate the transition from the introduction to the main body of the piece.

Free Musical Actions are parts of the musical surface which can be seen as actions by their relation to the musical context surrounding them. Their *meaning* or import can only be inferred as part of a complete musical interaction; this is the subject of the bulk of this chapter. These are the moments when the drummer introduces a new rhythm, and the bassist starts filling in the gaps; when a new melody gets passed around all the members of an improvising ensemble; when the backing section adopt ideas from a soloist's playing.

In some cases, the line between these different kinds of actions is not entirely clear; James Brown's shouting could be considered instructions outside the musical context, and samba *mestres* will use a combination of whistles and hand signals to work with their groups. In particular, the dividing line between musical actions which are understood through convention and those which are understood through some more general musical understanding may be fuzzy. The desired distinction is that conventionalised actions are defined by particular configurations of notes — specific forms of playing which can, in general, be made *explicit*. Free musical actions can only be understood in the context of a particular musical interaction, and deal with the relationships between the playing of the players rather than any specific phrases or material.

This chapter deals almost exclusively with free musical actions; conventionalised actions are style, group and piece specific, and can generally be described using standard methods, e.g. a rule based approach. Extramusical gestures are a sufficiently large topic that their existence is presumed, but not explored formally here[1].

6.2 Musical Agent System Description

This section develops a formal description of a musical interaction, and adds some assumptions to make subsequent reasoning clearer. It constructs the bottom layer of the representational stack — the musical surface — in a manner which works for the entire group of agents.

6.2.1 State Description Overview

Chapter 4 gave an architectural overview of an agent system. This section attempts to formalise certain components of this system.

A musical interaction can be modelled as a set of agents A_i and an environment E in which they are situated. These agents may be playing music, or simply listening; equally, they may be virtual or human.

It will be assumed that music is transferred at the level of the "musical surface", a term adopted from Lerdahl and Jackendoff [1983], where musical events are abstracted from the acoustic waveform. These events are portions of sound which are perceived as being single entities - common examples being notes, trills, cymbal crashes and so on. The rest of this discussion will focus on musical events which can be modelled as notes - having onset, pitch, duration, intensity and timbre. However, this is a convenience,

[1]However, they are discussed elsewhere in this thesis in the context of a musical agent system.

and should not be taken to mean that other types of events are not possible. Small parts of the musical surface will be denoted m.

For this analysis, a minimal set of agent characteristics will be used; these are not intended to give a complete description of the way which real agents make music, but to lay out the capabilities which are vital to the model being constructed. A minimal set of capabilities is used in order to maximise the range of possible agents which fit into this model. The necessary components are:

G**: Generation Processes** Processes which generate music and transmit it to the environment; this includes all of an agent's knowledge about music, both general and style specific, and all of the agent's abilities to produce music based on that knowledge. Different agents will have different sets of both musical knowledge and musical capabilities.

F**: Analysis Processes** Processes which receive input from the environment and analyse it; again, this includes individual skills and knowledge for this particular agent. As well as directly perceiving events, an agent will process the incoming data, in order to extract features — a high level representation of the music. Music is analysed by humans in a multitude of ways, from low-level concepts such as tactus and volume up to "groove", emotional impact and overall form. This is represented by the feature variable F, which is a set of features which the agent can extract from the musical surface.

C**: Context** A representation of what is currently happening in the interaction. This is a high level description of the musical interaction, based on the features produced by F, and is used for deciding what to play and interpreting the playing of other agents.

O**: Other Agents** The agent has beliefs about the other agents - their capabilities and intentions. As an agent plays with others, it becomes aware of them. There are many forms this can take; at a basic level, one might notice fellow musicians, and what instruments they are playing. During the course of a piece, the capabilities of the other musicians may become apparent, which can be used in deciding what to play next. Over the course of several pieces, or a musical career, information about the people one plays with will be built up, so there is a of model for all the partners in a given interaction.

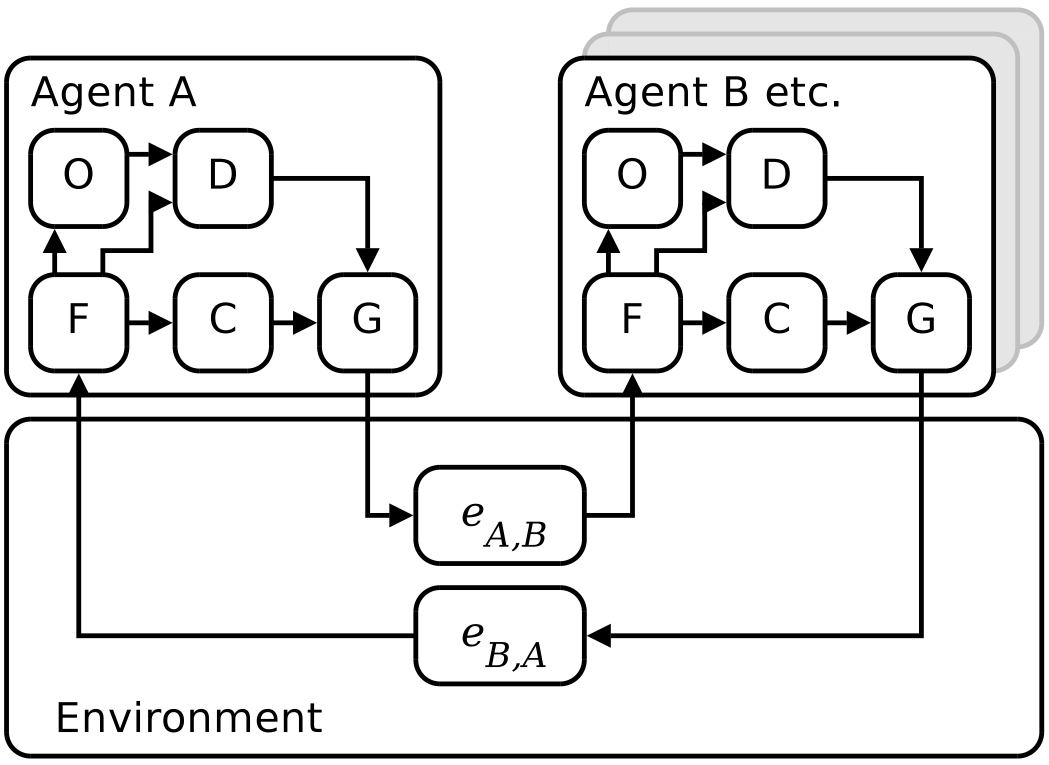


Figure 6.2: Overview of the formal agent model

D: **Decision Making** At some level, the agent must integrate all of this information, and use it to influence the generation of music. The decision making component has access to the context of analytical features and the model of other agents, and can use these to influence the workings of the generation processes.

Finally, the environment E is responsible for:

- locating the players in a space,
- transmitting music from one player to another,
- spatial effects on the music played.

The effects of the environment will be modelled as a set of transfer functions E, where $e_{A,B}$ is the effect of the environment on the music produced by agent A when it is heard by agent B.

These capabilities of agents and their environment are represented in Figure 6.2.

Finally, there are some properties and capabilities which the agents are likely to have, which are not included in this model:

Name	Symbol	Meaning
Next	$\bigcirc p$	p will be true in the next timeframe
Finally	$\diamond p$	p will be true at some point in the future while $\diamond p$ holds
Globally	$\square p$	p will be true for the entire future
Until	$p \mathcal{W} q$	p will be true until q, at which point it need not be true anymore. This is the weak version of until, which allows for q never becoming true

Table 6.1: Temporal Logic Operators

- Extramusical events will often be present; hand signals, eye contact, body movement etc. are all part of normal human musical interactions. As nothing is being said about the physicality of the agents involved, these extramusical events are ignored in this model.

- Most forms of music will involve some referent, or score, even if it does not specify all the notes which are to be played. This is some kind of schema or plan which the agents playing together can use to coordinate their musical activities. The score need not be entirely static and predetermined; in a “jamming” situation, the purpose may be to create a new referent to be used as the basis for further work. In extemporaneous composition, the referent is created through realising it [Sarath, 1996]. Although it is likely that agents will be using some form of score, this is not a necessary part of this model, and hence will be left out.

- agents are likely to have some set of goals, intentions or attitudes which influence the way in which they interact with other agents; rather than being explicitly modelled, it is assumed that these are taken care of by the deliberation module.

6.2.2 The Mechanics of Playing Music

To make a formalisation of the system easier, a formulation based on linear temporal logic is used. Emerson [1991] describes time as an ordered set of discrete states. Here, it is important to acknowledge that there is still some degree of temporal structure within each of these states, so continuous time is divided into a series of contiguous chunks, or timeslices. These divisions are not related to any musical notions — they have nothing to do with beats, bars or phrase groupings — but would be measured in seconds (or some other *physical* measure of time). The times of division are indexed — $t^0, \ldots t^n$ — as are timeslices — r^0 indicates the span of time in the range t^0 to t^1.

This allows us to talk about the music produced by agents in discrete units. These will be called *fragments,* such that f^n is the chunk of music played in r^n. Since the size and position of timeslices are not related to musical features, it is probable that notes and other objects on the musical surface will overlap fragment boundaries. When it is necessary to talk in more musical units, m may be used, such that ${}^a m^b$ is the set of fragments $f^a \ldots f^b$ which contain a particular musical structure. In general, since the idea is not dependent on the indices, m will be used alone. Each m is implicitly indexed, however; they represent pieces of music played *at a specific point in time*, not just the musical object which was played.

To formalise the temporal mechanics of the system, Temporal Logic operators are used [Emerson, 1991]. These can be seen in Table 6.1 (page 96); the presentation is based on symbols rather than letters, as in [Orgun and Ma, 1994].

If agent A plays a fragment f_n of music in timeslice n, we can express this as:

$$Plays_A(f^n)$$

This is true at t^{n+1}, and at all later time points it is true that A has played this fragment:

$$Plays_A(f^n) \rightarrow \Box Played_A(f^n)$$

This can also be applied to larger pieces of music, such that:

$$Plays_A(m) \rightarrow \Box Played_A(m)$$

In the physical world, all playing happens within an environment, which transmits the sound between players and listeners, with varying degrees of alteration. As sound travels from one player to another, different frequencies are attenuated by different amounts, the overall volume decreases, and an amount of time passes. The listener will also hear the results of the sound interacting with the environment — reverberations, early reflections, echos and so on.

When creating a virtual, networked ensemble, there are a different set of possibilities and constraints on the effect of the environment. Since the environment is simulated, it is no longer passive; this means that action must be taken to allow all the musicians to hear each other - otherwise, they will be playing in isolation.

The effect of being situated in an environment with other agents is that after one agent has played something, the others will all hear it. Hence, (for each agent $B \neq A$):

$$Played_A(m) \rightarrow \Diamond Heard_B(e_{B,A}(m))$$

In other words, at some point in the future, B hears a transformed version of what A has played, with $e_{B,A}$ defining the effects of the environment for music travelling from A to B. This formulation does not define at what point the other agents hear A's playing; it may be many timesteps before the sound reaches B.

For the purposes of this analysis, it will be assumed that

- the environment is such that it does not distort the sound.
- the agents are situated such that they always hear playing in the next timestep.

Bearing in mind that m relates to a specific piece of music played at a specific time, so that once B has heard it, it is always true that B has heard it, the simplification is then:

$$Played_A(m) \rightarrow \Box Heard_B(m)$$

6.3 Describing Music

This section develops the second layer of the representational stack — the level of features — from the point of view of a single agent.

Although there is no clear dividing line between the perception of music and its processing into higher level structures, for the purposes of this model the assumption is made that all agents perceive music at a level similar to the “musical surface” used in the GTTM Lerdahl and Jackendoff [1983]; that is, music is segmented into discrete events, at the level of notes, percussive strikes, glissandi, trills etc. In this analysis, the musical surface will be represented by a logical notation, which represents these discrete events as objects whose types correspond to the type of event, with whatever parameters are necessary to describe the event. An example would be:

```
(Note pitch:c4 onset:1.0 duration:  1.5 intensity:  0.7)
```

Even this is not without problems, as different musical ontologies are necessary to describe different musical domains; however, this is a reasonable (and common) assumption for this type of analysis.

In order to build up a model for expressing higher level musical structures, we must allow for the following properties of musical situations:

- there will be several descriptions which could be applied to the same piece of music - for example a single note may well have rhythmic, harmonic and melodic functions; in my opinion, this polysemic nature is one of the most interesting properties of music.

- the set of descriptions used will vary with the type of music and the skills and interests of the performer and listener, so not only may different agents have different capabilities, the range of potential capabilities cannot be known ahead of time.

It is also necessary to be able to calculate these descriptions and reason over their values in order to construct appropriate output.

The rest of this section deals with developing a framework whereby multiple analytic features can be attached to fragments of music, as long as some properties are maintained by the feature systems used.

6.3.1 Facets and Values

In order to capture the notion of multiple structures being attached to a piece of music, the notion of *facets* will be introduced; these are descriptive classifications for certain aspects of the musical surface. An analysis procedure carries out the task of attaching a specific *value* to a *facet*; this mapping will be termed a *descriptor*, and the process an *analyser*.

A value for a facet should be thought of in set-theoretic terms - it defines a certain subset of all possible pieces of music - so an analysis procedure should (in general) find the smallest possible subset to which a given piece of music belongs. Before progressing further, it should be stated that:

- descriptors are essentially perceptual objects; a given set of descriptors only accounts for a *possible* analysis of the music; they are subjective and not necessarily complete.

- in the coming examples, the descriptors used may not seem to capture the essence of musical knowledge about the subject; this should be taken as a failure of those particular descriptors, rather than an issue with the theory - simplistic descriptions of music have been purposefully used to make the logical structure as clear as possible. How this scales up to more musically complete descriptors will be addressed separately.

- facets will be considered independently, although in general there may be some relation between the values assigned to certain aspects.
- few demands are made on the structure and type of values allowed - this is designed to make sure the system is extensible to different musical features, which will require their own treatments.

An informal notation will be used here to represent these descriptors as tuples; as an example, a very simple descriptor would be (Time-Signature, 4/4), which delimits the set of possible musics where the signature is 4/4. Values can potentially be complex objects, for example: (Chord-Sequence, (C,C,F,G)) delimits the set of musical fragments based on a four bar I-I-IV-V progression in C. In general, it is not important what form these values take, so long as certain properties hold, as described in the next section.

6.3.2 Relations between values

In order to allow discussion of relations between values, a system based on Concept Lattices is used (see e.g. Ganter and Wille [1997]). Formally, a lattice is a partially ordered set (poset), where every pair of elements has a unique *supremum*, or meet, and *imfimum*, or join; a semilattice is similar, but only includes one of the two operations (meet or join).

The definition of a concept lattice is used as a starting point for defining relations between values. A concept lattice operates over a set of objects O and a set of attributes A, and a concept is defined as a pair (O_i, A_i), where:

- $O_i \subseteq O$
- $A_i \subseteq A$
- every object in O_i has every attribute in A_i
- for every object in O that is not in O_i, there is an attribute in A_i which that object does not have
- for every attribute in A that is not in A_i, there is an object in O_i which does not have that attribute

Since we are dealing with the set of all possible musical fragments, we will avoid enumerating O, or any O_i.

Name	Symbol	Meaning
Any	$\odot$	No value specified for this object or part of object (c.f. $\top$)
None	$\ominus$	This value has been overspecified - no consistent assignment can be made (c.f. $\perp$)
Sub	p $\subset$ q	p is subsumed by q; or p is a weaker version of q (c.f. $p\langle q$)
Combine	p $\otimes$ q	The result of combining the two sets of specifications; also known as the *meet* or *infimum* (infix notation)
Combine	$\otimes$ ($p_1 \ldots p_n$)	The *meet* for a set of values (prefix notation)

Figure 6.3: Lattice value operators

The property of most relevance here is the partial ordering of concepts, which makes this structure a partially ordered set, and is defined as:

$$(O_i, A_i) \leq (O_j, A_j) \underset{def}{\leftrightarrow} O_i \subseteq O_j$$

(which is equivalent to $(O_i, A_i) \leq (O_j, A_j) \leftrightarrow_{def} A_j \subseteq A_i$)

For the purpose of this theory, two special values are introduced:

$\odot$ may be used to signify that no value is specified for that facet. This corresponds to to the set of all possible musical fragments; it is analogous to the concept $\top$ used in Description Logics (e.g. Baader [2003]).

$\ominus$ is the empty set, and is encountered when values are combined such that no pieces of music can satisfy the resulting condition. This is analogous to $\perp$ in Description Logics.

In order to be considered a possible value system for a facet in a Musical Act system, the concepts and relations shown in Table 6.3 should be present. It should be noted that although an operation for *meet* is required, there is no corresponding requirement for a *join* (or *supremum*). This has been relaxed as joins have not been necessary, so allowing a greater range of possibilities is worthwhile. This means that the structures are *semilattices*.

In order to characterise the relations between different values in a way which is independent of the particular facet under question (necessary for a general theory), only the properties which are demanded by the definition of a value system may be used. From these several relations are chosen between a new value n and an old value o:

Same is as expected, $n = o$

Subsumes indicates that the new value subsumes the old, or that $o \subset n$. In graph terms, this means the new value is an ancestor of the old.

Subsumed indicates that the new value is subsumed by the old value, or that $n \subset o$, i.e. n is a descendant of o, and describes a smaller set of musical values.

Alter indicates that n and o share a common ancestor a, or in lattice terms that there is a value c such that $n \subset a$ and $o \subset a$.

Disjoint indicates there is no common ancestor for n and v (except $\odot$, which is an ancestor of every node), or that their *meet* is an empty set ($n \otimes o = \ominus$)

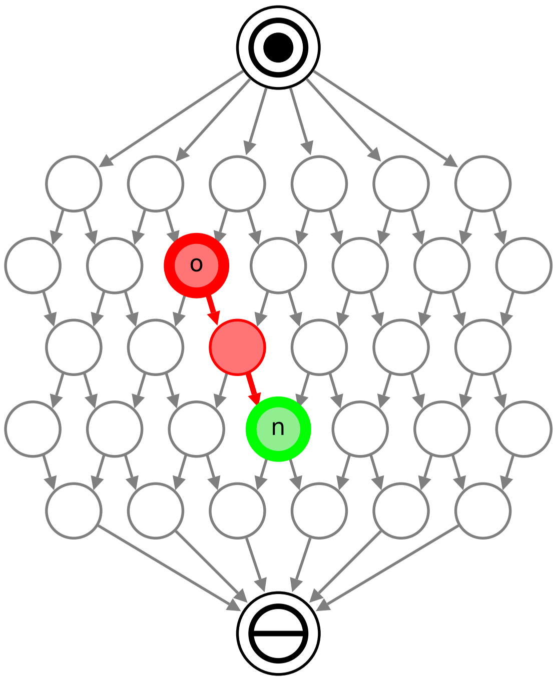
These are shown diagrammatically in Figure 6.4.

6.3.2.1 Examples of musical lattices

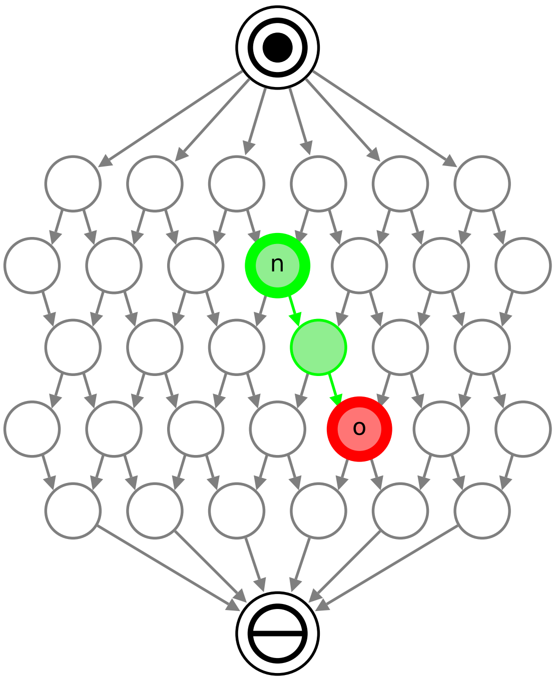
At this point, some examples of different value lattices and the computation of relations between their members are illustrative. Again, it should be understood that these are *possible* treatments of these value-systems; much of the richness of possible representation is ignored in favour of giving a concise demonstration of the structures under discussion.

Chord Roots can be treated simply, with a lattice shown in Figure 6.5. Here, any given chord can have only one root, taken from the set of pitch classes. Attempting to combine two different roots results in an incompatible value, as does combining anything with $\ominus$, while combining anything with $\odot$ leaves the value unchanged. For example:

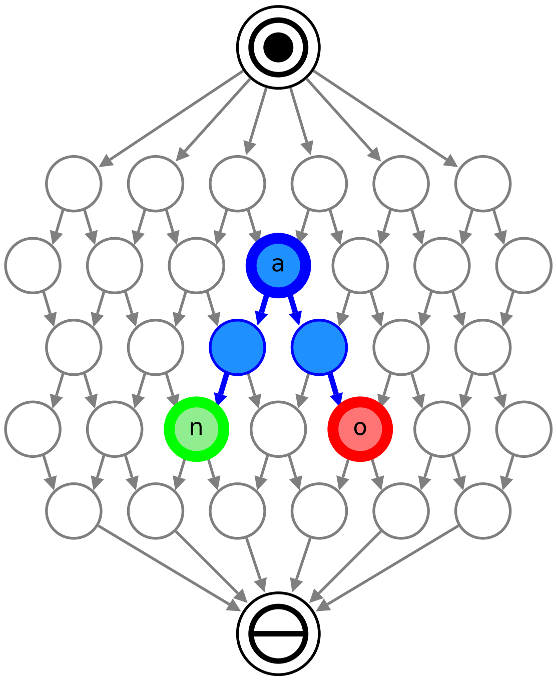
- $Bb \otimes Bb = Bb$
- $Bb \otimes D = \ominus$
- $Bb \otimes \odot = Bb$



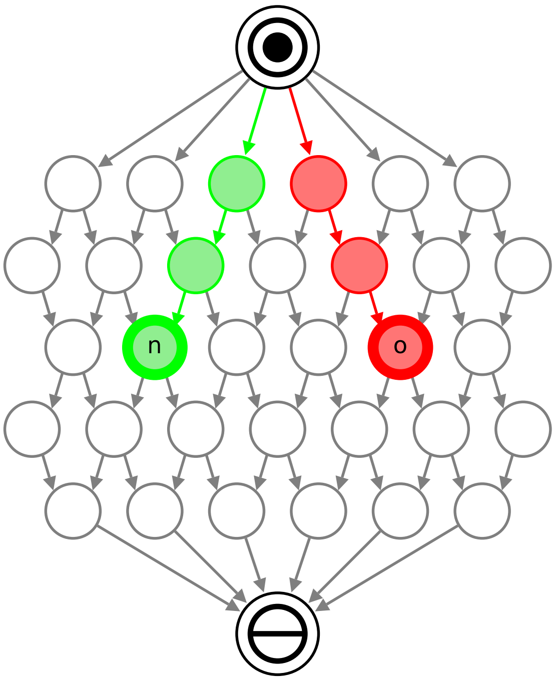
(a) Subsumed Concept Relation



(b) Subsumes Concept Relation



(c) Altered Concept Relation (*a* is shared ancestor)



(d) Disjoint Concept Relation

Figure 6.4: Illustration of different concept relations between an old value *o* and new value *n*

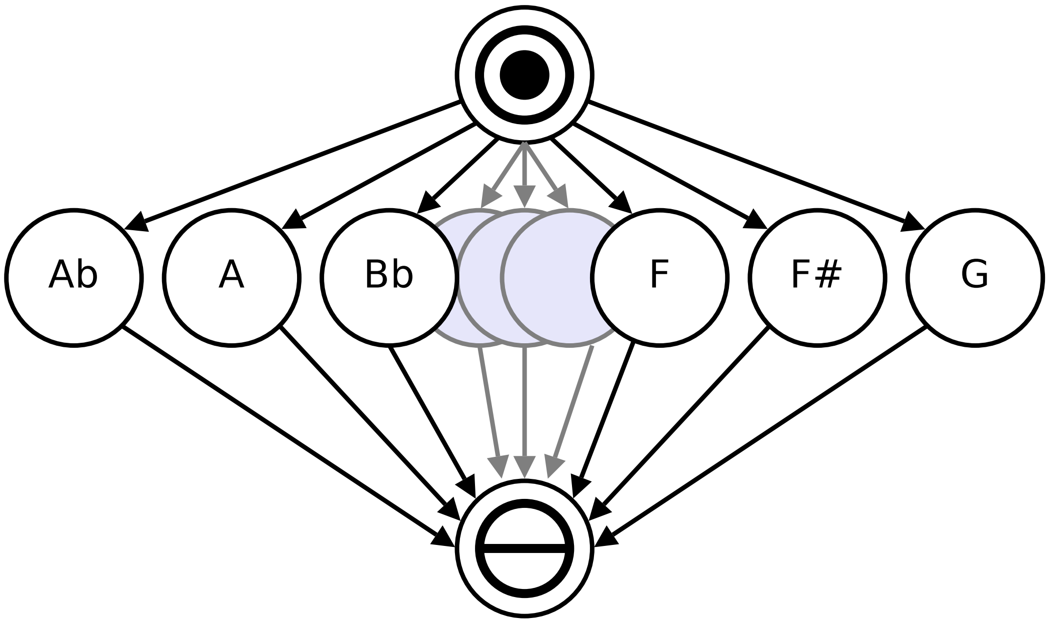


Figure 6.5: Chord Root Lattice

- $Bb \otimes \ominus = \ominus$

For this lattice, the only relationships which are possible are `SAME` and `DISJOINT`.

4 bar sequences of chord roots uses the previous lattice of chord roots to fill in values for each bar in a 4 bar sequence — (see Figure 6.6 for a partial exploration of this space). The top value in the lattice is a completely unspecified sequence; each downward jump specifies one more root for one slot of the chord. When sequences are combined, each slot of the new value takes on the value formed by combining the equivalent slots in the two parents. After 4 downward steps, we have the set of all possible 4 chord sequences, and after this, each successive combination will result in one or more of the slots becoming overspecified, and taking the $\odot$ value. Given two values, v_a and v_b, the relation between v_a and v_b is calculated as follows:

- if $v_a = v_b$, the relationship is `SAME`.
- otherwise, $v_c = v_a \otimes v_b$ is calculated. Each slot of the chord sequence is combined with its opposite number in the other chord sequence, using the rules given above for combining chord roots.
- if $v_c = v_a$, then v_a is `SUBSUMED` by v_b — since the definition of v_b can be narrowed to produce v_a, v_a must describe a smaller set of musical output.

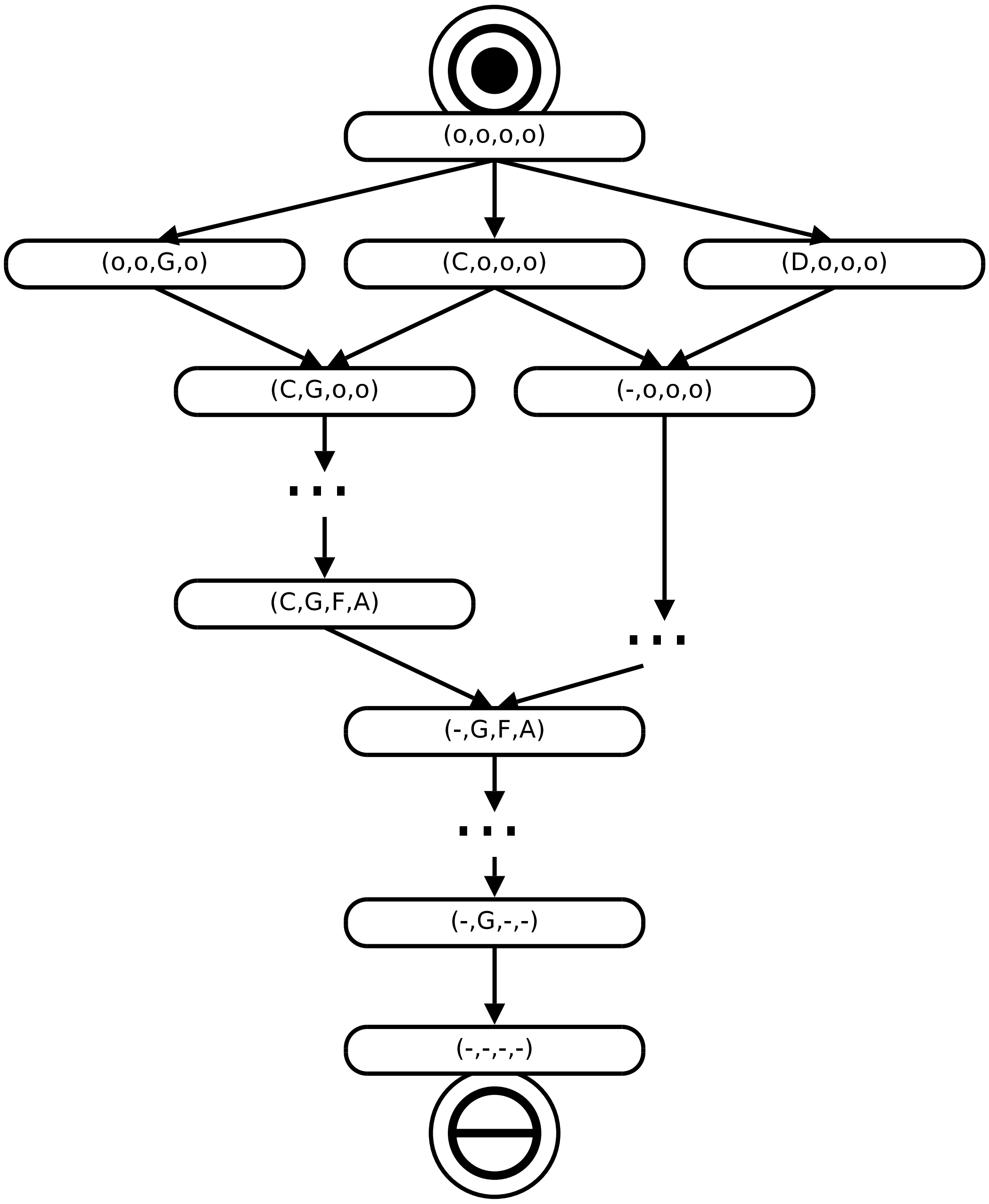


Figure 6.6: Partial lattice of sequences of chord roots

- similarly, if $v_c = v_b$, then v_a `SUBSUMES` v_b.
- otherwise, if any of the slots of v_c contain a value which is not $\odot$ or $\ominus$, the v_a `ALTERS` v_b — the common ancestor can be constructed by replacing every $\ominus$ in v_c with $\odot$.
- finally if every slot in v_c is $\ominus$ or $\odot$, then v_a is `DISJOINT` from v_b.

For example:

- (C,C,G,F) is `DISJOINT` from (D,D,F,Bb).
- (C,C,G,F) is `SUBSUMED` by (C,C,$\odot$,$\odot$).
- (C,C,G,F) `ALTERS` (C,C,G,Bb), with a common ancestor of (C,C,G,$\odot$).

Chords with extensions up to the 7th A chord is a complex object, but in jazz terminology, it can be broken down into a root, followed by a number of extensions[2]. a simplified representation of this is $\{r/\quad,3/\quad,5/\quad,7/\quad\}$, with the values for $r,3,5,7$ filled in as necessary. Roots come from the set of note names, thirds may be major or minor, fifths are assumed to be perfect for simplicity, and sevenths may be major or minor. Extensions may also take the values $\odot$ and $\ominus$ (represented by 'o' and '-' respectively on the diagram). A value of $\odot$ means no extension is specified for this chord (e.g. C major does not specify a 7th), while $\ominus$ is the result of combining two chords with one or more differing extensions. (see Figure 6.7) for a representation of part of this lattice.

6.3.3 Analysis Procedures

An analysis procedure A_f for a particular facet f can be defined as a function from fragments of music M to lattice values v, constrained to be on the lattice L_f associated with that facet. That is:

$$A_f : M \rightarrow L_f$$

Once these values have been produced, they are notated as tuples containing the facet and value, i.e. (f_i, v_i), to allow values for different facets to be distinguished from each other.

It is generally assumed that an agent will have a set of analysers which it can apply to new music; the set of facets over which these analysers act is notated F.

[2]More usually, a chord is thought of as a triad plus extensions, but root+extensions is a good fit for this lattice representation.

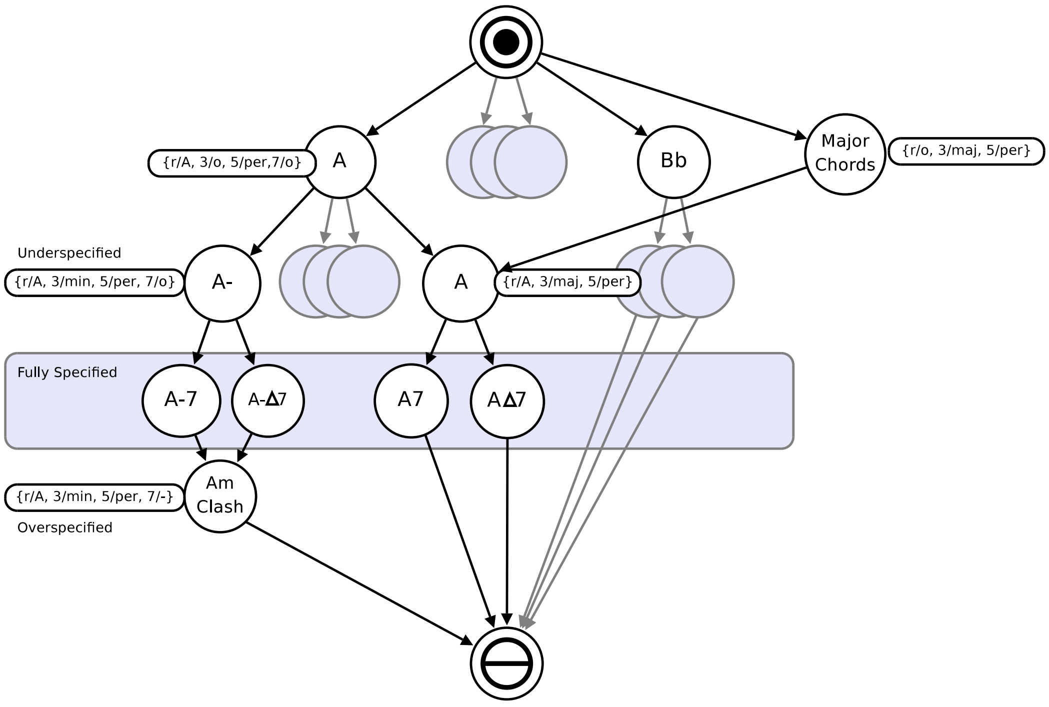


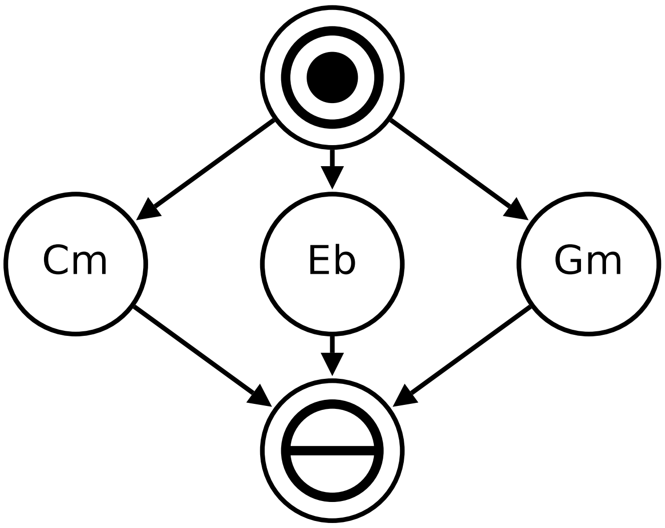
Figure 6.7: Partial lattice for chords with extensions up to the 7th

There is no general restriction on the manner in which the musical surface is mapped onto lattice values; the restriction is on the form which the lattice values may take — that is, the relations between lattice values must be calculable. In order to make this clear, two chord analysers are detailed below. The lattices for these analysers are shown in Figure 6.8.

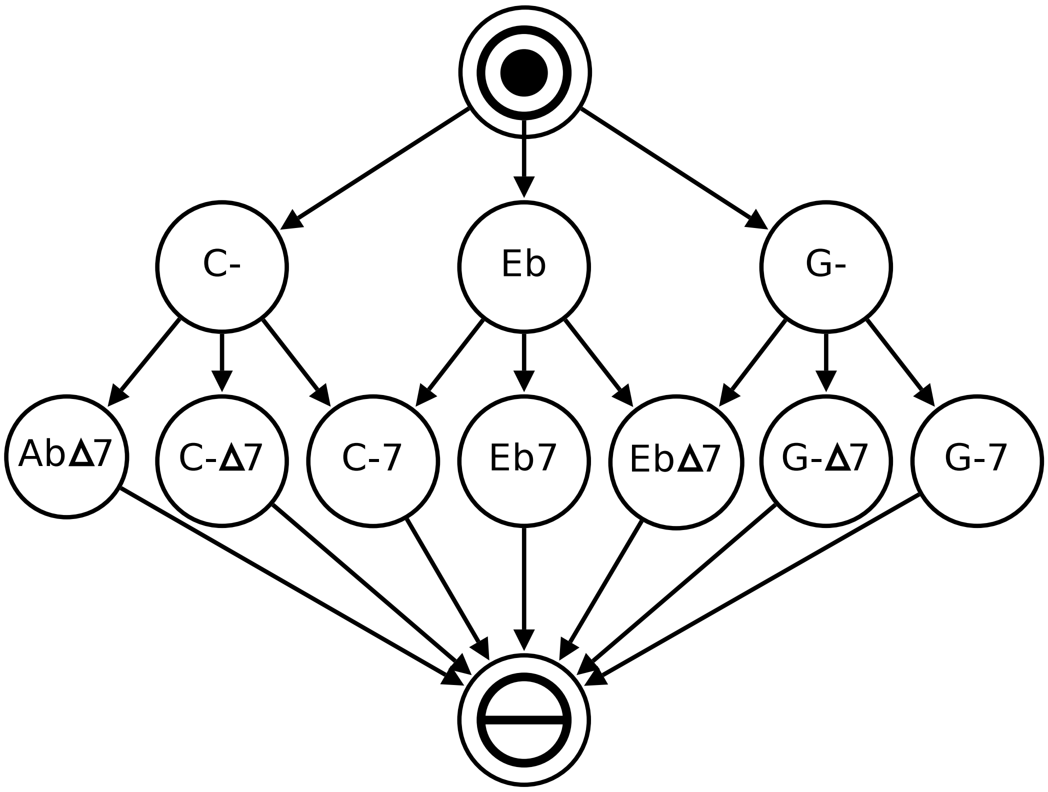
The first analyser — $A_{nursery}$ — is a "Nursery Rhyme" treatment of chords, where the lattice simply comprises a node for each major and minor triad (Figure 6.8(a). In this lattice, only two relationships are possible - `SAME` and `DISJOINT` as all nodes are direct descendants of the $\odot$, and have only $\ominus$ as a child.

A_{jazz} takes a richer, jazz oriented approach. Here again, we start with a node for each triad, but this is now augmented with nodes for all the major and minor seventh variations of these chords. Node relations are worked out as follows:

- for each node, its parents are any nodes which can be created by removing a single note from the chord.
- a node's children are any nodes which can be constructed by adding a single note to the chord.
- using these parent/child relationships, the full range of relations can be com-



(a) Nursery Rhyme chord analysis



(b) Jazz chord analysis

Figure 6.8: Two different lattices for analysing chords (showing chords related to C minor, Eb major and G minor)

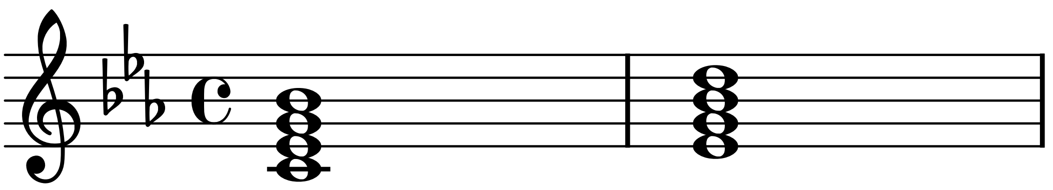


Figure 6.9: Example music for analysis

puted; a node subsumes all of its descendant nodes, and is subsumed by all of its ancestors; it is an alteration of nodes which share a common ancestor, and disjoint from those that do not.

A diagrammatic representation of this is shown in Figure 6.8(b).

Now consider the reaction of these two analysers to the piece of music shown in Figure 6.9, which contains two chords: C,Eb,G,Bb followed by Eb, G, Bb, D. The analysers are asked to calculate the values for each of the chords, and the relation between them.

$A_{nursery}$ cannot fully represent the C-7 chord, given that its universe consists only of three note triads. It has two possibilities — Cm or Eb — and would need some way to determine which of these was more appropriate. There is a third possibility, which is that it would refuse to classify the chord, but it will be assumed that it attempts to locate every input *somewhere* on its lattice, even if the fit is not exact. It is likely that a well designed analyser would choose Cm in this case. The following chord is analysed in a similar manner as Eb, When computing the relation between them, it can simply note that they are different nodes, and say that they are disjoint.

A_{jazz} can classify both of these chords exactly on its lattice, as C-7 and Eb7 respectively. Furthermore, when computing a relation between them, it can determine that Eb is a parent to both chords — since adding a C to an Eb chord gives C-7 and adding a D to an Eb chord gives Eb7 — and hence that moving from C-7 to Eb7 is an alteration.

There is then the matter of whether this ability to capture extra relationships between values and give richer descriptions of music makes A_{jazz} a better analyser, and the response is that it is up to a particular agent to decide which analysers are most useful for the style of music which it is playing — whether C-7 is seen as an alteration of Eb7 is dependant on the theory within which one is working, and different approaches will be appropriate for different musics. This is important to the generality of this model, as it means that any musical theory can be used so long as it can be represented as a lattice of values and an analysis function from the musical surface to these values.

6.4 Understanding other musicians

This section expands the featural level to take into account the different agents involved in an interaction, and what they can deduce about each other's musical beliefs. Imagine agent A plays a chunk of music

$$Played_A(m)$$

It is useful to be able to talk about what can be understood from this, and what one agent can expect the others to have understood. This section develops three concepts related to this: current values, musical context and common acceptance.

6.4.1 Extracting values

There are many features of this chunk of music which can be deduced by a musician or musical analyst; for this model of a musical agent, each facet for which the agent has an analyser results in a descriptor, e.g. (Tonality, C minor), (TimeSignature, 4/4) etc. Each chunk of music played has many of these properties. The symbol $\Rightarrow_A$ is used to mean "A can extract the property[3]", so if A has an analysis procedure for facets $F = \{f_1, \ldots, f_n\}$:

$$m \underset{A}{\Rightarrow} \{(f_0, v_{0m}), \ldots, (f_n, v_{nm})\}$$

In other words, for each facet $f_i \in F$, A can extract a value v_{im}.

These properties are not tied absolutely to the musical surface - rather they are a product of the interaction between an analyser and the surface, and different analyses could produce different descriptors. Since there are a potentially infinite range of analysis procedures which give rise to facet-value pairs, there is no objective, complete set of all the possible descriptors which could be created from a fragment of music, and the production of descriptors must be discussed in the context of an entity which can create those descriptors.

Each agent in the system has a set of analysis procedures, each of which can derive a value for a particular facet when given a piece of music. The formulation is then that when agent B hears agent A play a chunk of music, B believes that A has expressed[4]

[3]Property is used interchangeably with descriptor here.

[4]*Expressed* is used in a weak sense here, to mean that those properties could be perceived in the music - it does not necessarily mean that they were *intended*.

all the properties that B can derive from the music:

$$Heard_B(Played_A(m)) \wedge (m \underset{B}{\Rightarrow} D) \quad \rightarrow \quad Bel_B(Expressed_A(D))$$

where $D = \{(f_0, v_{0m}), \ldots, (f_n, v_{nm})\}$ for the set of descriptors which B can extract from m. It should be noted that this set of descriptions may be entirely different from whatever representation (or process) A used to create the music.

6.4.2 Temporality and Current Values

The extraction of values discussed above is outside of any temporal framework; it is simply the method of determining what features are embodied in a fragment of music. Since music happens within its particular temporality, this extraction process must similarly be embedded in a temporal structure. As previously noted, not all analysis procedures work to the same timescale; some that deal with large scale musical structures such as chord sequences will take several bars to have an idea of what is happening, while others which are close to the musical surface may react almost immediately. In order to abstract some of the complexity of these temporal mechanics, it is useful to capture the notion of what an agent is doing "at the moment". There are two notions of "at the moment" we can use here, which will be termed CV^I and CV^L (*I*nstantaneous or *L*asting):

CV^I says that current values hold only while they are being maintained — that is, an agent is only counted to be outputting music which embodies a certain value after timesteps in which that value can be extracted from its output. More formally:

$$Expressed_A((p_c, v_c)) \rightarrow \bigcirc CV^I_A(p_c, v_c)$$

CV^L says that current values hold until the agent produces a new value for that facet:

$$Expressed_A((p_c, v_c)) \rightarrow CV^L_A(p_c, v_c)\,\mathcal{W}\,(Expressed_A((p_c, v_d)) \wedge v_d \neq v_c)$$

It should be noted that analysers should maintain their output for the entire time that output embodying their value is received. To use an example, a chord sequence analyser, once it has determined that everyone is playing a 12 bar blues, would continue to output the value representing this while there was harmonic playing which was based on this blues. In the drum solo, however, it would stop, and it is at this point that CV_I and CV_L would have different values; CV_I would say "there is no chord sequence", while CV_L would assume that everyone was still implicitly playing the blues, since nothing has been heard to contradict this.

At this stage, it is not clear which formulation is more useful or representative of real situations, so from here on CV will be used to mean either formulation, with the choice of which to use being left as an implementation detail of a particular system.

6.4.3 A state indexed description of music

In order to hide some of the temporal complexity from our analysis, a state based description of the musical interaction is introduced. Looking at a single facet of analysis, the description is constructed as follows:

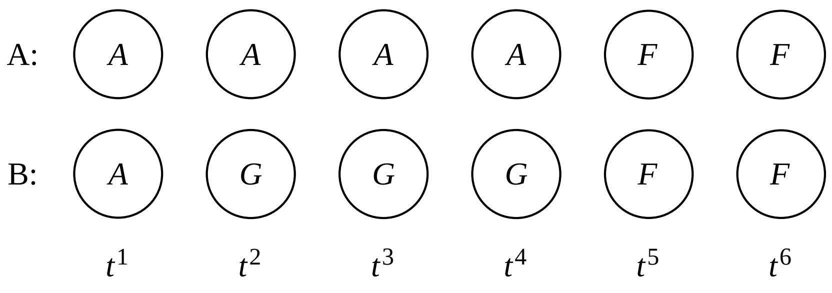
- each timeslice, a new fragment of music is produced, received and passed to the analyser, so the current value for that facet is updated to a new value v; this is timeslice indexed, so at time t^n, the analyser will produce v^n. As an example, Figure 6.10(a) describes the output of two agents (A and B) using an analyser which extracts the root of the chord the agent is playing.

- the sequence of values produced will contain many repeats, especially if the size of timeslices is small. In order to avoid having to list repetitive sequences, the notion of state based time is introduced. s_n now refers to the entire range of time for which the current value was v_n — that is, states change when current values change rather than being bound to timeslices. Properties which are state-indexed use subscripts for their index, while those which are timeslice indexed use superscripts.

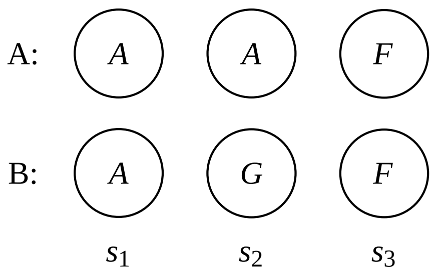
- when a set of values which are changing over time is considered — for example, the output of several agents for a single facet — a new state is defined every time one of the values changes. Figure 6.10(b) represents the same sequence of output as Figure 6.10(a) in a state indexed manner.

6.4.4 Musical Context — the Musical Now

In informal language, the musical context, or “musical now”, is intended to capture “what all the musicians are doing at the moment”. Although this may seem simple on the surface, the fact that music is a process which unfolds through time means



(a) Timeslice indexed description



(b) State indexed description

Figure 6.10: Timeslice and state indexed descriptions of musical interactions

that at any given point, describing the musical situation must certainly take into account events which are in the past, and may well refer to possible events in the future. Some descriptions may span several bars "he's playing the melody from *Summertime*", "they're jamming on a 12 bar blues", and some may be far more temporally specific "she's playing an Eb".

In the state indexed description of music, the musical context can be simply represented as the set of all values — for all agents, for all facets — which are present in the current state. More formally, for an agent x in a group of agents A, with a set F of facets it can analyse, the context is the conjunction of all relevant facet-value statements:

$$Context_x = \bigwedge_{a \in A} \bigwedge_{f \in F} CV_a(f, v_f)$$

6.4.5 Common Knowledge and Musical Common Ground

So far, these formulations work for a single agent analysing the output of its peers. In order to reason about what actions to take in a particular context, it is useful to be able to draw some conclusions about the beliefs of other agents. So far, we have assumed that all of the agents can hear each other's output, but nothing has been assumed about what any agent derives from the output of the others. The notion of common knowledge is useful here,

> it is common knowledge that ϕ among a group of agents iff all know that ϕ, all know that all know that ϕ, all know that all know that all know that ϕ, etc.

This will be notated $CK(\phi)$. Using the assumption that every agent can hear the output of every other, and that this fact is common knowledge, it is clear that the playing of any agent is common knowledge (in the next timestep):

$$Played_A(m) \rightarrow \Box CK(Played_A(m))$$

as is the fact that this has been heard by every other agent (for all $B \neq A$):

$$Played(A,m) \rightarrow \bigcirc CK(Heard_B(Played_A(m)))$$

It is, as noted before, easily possible to imagine situations in which it is not the case that all of the musicians hear each other, but this covers a wide range of "normal" musical situations, and is in most cases a state which is desirable even if not attained.

If an agent believes that another agent can analyse a certain facet of music, and will produce the same values for that facet, it will believe that this agent will derive the same values from the music it hears:

$$\begin{array}{rl} & Bel_A(CanAnalyse_B(f)) \\ \wedge & Heard_A(Played_C(m)) \\ \wedge & Bel_A(CV_C(f,v_f)) \\ \rightarrow & Bel_A(Bel_B(CV_C(f,v_f))) \end{array}$$

Hence, given a set F_s of facets for which A believes B will analyse in the same way:

$$Bel_A(CV_c(f,v_f)) \rightarrow Bel_A(Bel_B(CV_c(f,v_f))) \qquad \text{(for } f \in F_s)$$

Now, suppose that A believes there is a set F_c of facets which every agent can analyse, that all agents will arrive at the same values for these facets, and also that these facts are common knowledge. It follows then that A believes the values which can be deduced for these facets are common knowledge:

$$Bel_A(CV_x(f,v_f)) \rightarrow CG_A(CV_x(f,v_f))) \qquad \text{(for } f \in F_s, x \in Agents)$$

This is defined as *musical common ground* — the set of values which an agent reasonably believes to have been extracted by every other agent, and hence to be common knowledge.

So, is the existence of sets of common descriptions a reasonable assumption? Much of music education, particularly music theory, is concerned with creating a common vocabulary for musical occurrences; in general, within any particular style, there are aspects of music which it would be taken as read that any skilled practitioner would understand. It would be assumed that people will agree on the time signature a piece is in, what the chord structure is for jazz musicians, what *taal* or *raga* is being used in Indian music etc.

There are many situations where these assumptions turn out to be false: people may disagree about what chords are being used, about whether a piece is in 6/8 or 3/4 etc., but they are reasonable assumptions to make within a given cultural context. Similarly, when musicians with different backgrounds get together, the shared set of features may be smaller; the marching band drummer may be insensitive to the varieties of “swing” employed by a samba band; the classical pianist unable to distinguish between the *ragas* employed by the harmonium player etc.

For these reasons, the set of features which any agent shares with another may dynamically change, both within a single interaction and over the course of many interactions as expectations are adjusted, new capabilities discovered and deficiencies exposed.

It is sometimes useful to think of common ground among a subset of the players: the beginning bassist may be happy to know the root of each chord being played, while the pianist and guitarist share an understanding of the extensions and passing notes they are using, and the drummer might have little knowledge of the actual chords used (see Figure Figure 6.11), but share a deep understanding of the rhythmic properties with the bassist. Here, there are several common grounds:

- between the entire group about what the time signature is, and what the structure of the piece is,
- between the bassist, guitarist and pianist about the basics of the chord sequence,
- between the guitarist and pianist of the complex harmonic material being used,
- between the bassist and drummer about the complex rhythmic material being used.

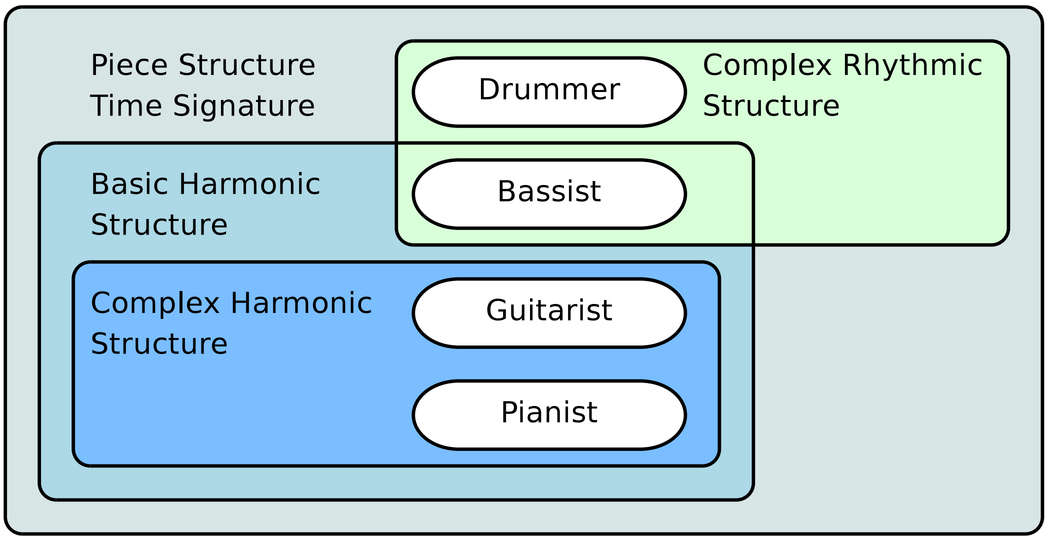


Figure 6.11: Example of different common grounds for a group of agents playing jazz

6.5 Actions within musical context

By defining the state of music, and the expectations or beliefs the agents involved might have about each other, the groundwork has been laid for discussion about actions within that framework. In order to do this, an assumption is made:

$A_{context}$: everything which is of interest to an agent about the musical interaction is contained in the context which it maintains.

Given this, in order for anything to constitute a free musical action from the point of view of an agent hearing it, it must involve a change to that agent's context, and hence must involve a new state in the state description:

A_{action}: all free musical actions consist of a change of state.

Turning this around, the converse assumption is that:

A_{state}: every change of state constitutes a musical action on the part of the agent whose value changed the state.

Both of these assumptions are discussed further in Section 6.5.7, but accepting them leads to the conclusion that analysing the transitions between states gives a picture of the intentional actions present in the interaction.

6.5.1 Actions and relations

In order to extract actions from the musical surface, new playing is related to the context in which it occurred — in other words, it is compared to the current values of the whole group at the time the action happens. Every time a new value is expressed by an agent, a new state is created. The action is then a combination of:

- the musical surface so far, including the part which constitutes the action,
- the state description of the musical interaction,
- previous actions,
- the agents involved in the interaction,
- the new value which has just been played.

This refers to a specific action, by a specific agent at a certain point in time in a specific interaction. Since this is such a specific event, it is not easily generalisable to other situations. Instead, a *musical action signature* is constructed, which consists of:

- the relations between the new value being played and the current musical context.
- relevant relations between values in the current musical context.

Figure 6.12 gives an overview of the relationships available, where arrows between states indicate relations which may be used.

6.5.2 Action signatures for two agents

In order to simplify this, consider the case of two agents playing together; agent A has introduced a new value for a particular facet, so the values which are available to characterise the act are:

- A's new value (a_{new}),
- A's old value (a_{old}),
- B's old value (b_{old}).

Since relationships are directional, there are six possible relationships to consider. However, since special significance is given to the new value played by agent a, relations are considered between:

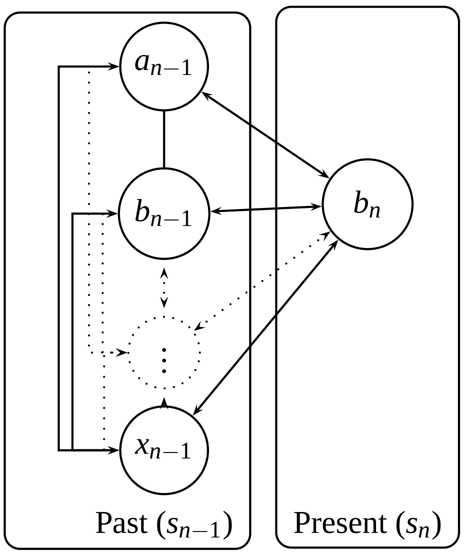


Figure 6.12: Available relationships for construction of a new value, with agents $a, b, \ldots, x$ from the point of view of b constructing a new value

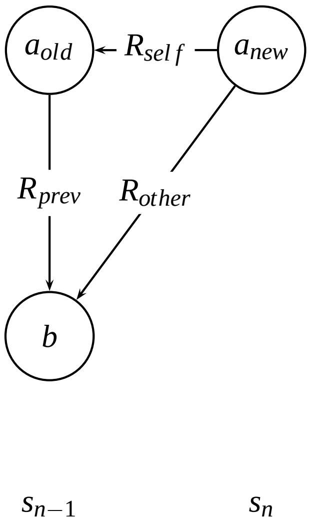


Figure 6.13: Relations between the values of two agents a and b, from the point of view of a constructing a new value

- a_{new} and a_{old},
- a_{new} and b,
- a_{old} and b.

(See Figure 6.13 for details)

Put together, this means an action signature is defined by a triplet of relations — from the previous set of $\{$`SAME,SUBSUMES,SUBSUMED,ALTER,DISJOINT` $\}$ — for values of a certain facet.

$$ActionSignature \underset{def}{\leftrightarrow} (R_{self}, R_{other}, R_{prev})$$

This does not entirely define the action; it characterises the relational aspects of it independent of the context it occurred in. If more context is needed in a particular application, a *contextualised action signature* may be used, for example including the agent carrying out the action, the agent the action is being calculated relative to, the facet in which this action is occurring and the time it happened:

$$ContextualActionSignature \underset{def}{\leftrightarrow} (\text{executor}, \text{relativeTo}, \text{facet}, \text{time}, (R_{self}, R_{other}, R_{prev}))$$

In the rest of the text, it will often be important to know which agent produced an action signature. In this case, it will be written $AgentName : (R_{self}, R_{other}, R_{prev})$, e.g. A:(`SAME,ALTER,DISJOINT`).

This still does not completely define the action; it does not include the musical surface, or even the part of it deemed to constitute the action; it is designed as a compact, transferable representation of the components of the action which are useful from the point of view of analysing interactions. The interaction can now be described using a series of these action signatures, as will be demonstrated in the following paragraphs.

6.5.3 The universe of action signatures

An action signature consists of a triple of relations, where each relation can take one of five values — `SAME,SUBSUMES,SUBSUMED,ALTER,DISJOINT`. At first glance, this would appear to give 125 possible values for an action signature. On further inspection, however, this is an overestimate; for example, it is not possible to have an action where R_{self} is `SAME` — there would be no grounds for calling it an action. There are other constraints on which relations can hold between three values, and this is the subject of an investigation in Appendix B. The final table of possible action signatures is given in Figure 6.14.

6.5.4 A worked example

A this point a worked example is useful, to demonstrate these concepts in action. This involves two agents, *A* and *B*, who are both jazz musicians, and looks at a chordal analysis of their output. The analysis is going to be conducted from the point of view of *A*, who is using the simple jazz chord analyser (A_{jazz}) from Section 6.3.3 to extract chords and their relationships.

At the start of the example, *A* is playing a C chord (i.e. C,E,G), while *B* is playing C7 (C,E,G,Bb), so *A*'s value `SUBSUMES` *B*'s. At some point, *B* decides to start playing $C\Delta7$ instead (C,E,G,B). This allows the calculation of three relations:

R_{self} is the relation between *B*'s new and old values. Even though *A* is conducting the analysis, when analysing the actions of others, R_{self} describes the relations between the *other*'s values. The relation is from CΔ7to C7, and is hence `ALTER` — they have a common ancestor in *C*.

R_{other} is the relation between *B*'s new playing and *A*'s current playing, so between CΔ7and C, and is hence `SUBSUMED`.

R_{prev} is the relation which held between *B*'s old value and *A*'s current value, i.e. between C7 and C, and is again `SUBSUMED`.

This means that from *A*'s point of view, *B* has executed a musical action with the signature B:(`ALTER,SUBSUMED,SUBSUMED`). It should be noted that *B* might have been using an entirely different method of analysis and generation — this analysis depends on *A*'s perceptions and capabilities, which may be entirely different to *B*'s. The exchange can be seen in Figure 6.15.

A longer excerpt of the same example is encoded in Table 6.2.

6.5.5 Choices

This theory of musical actions has a dual purpose: to describe the music played by agents in terms of high level actions, and to give agents a framework for making decisions about what to play.

$R_{prev} = \texttt{SAME}$

R_{self}	Allowed values for R_{other}
SUBSUMED	SUBSUMED
SUBSUMES	SUBSUMES
ALTER	ALTER
DISJOINT	DISJOINT

$R_{prev} = \texttt{SUBSUMED}$

R_{self}	Allowed values for R_{other}
SUBSUMED	SUBSUMED
SUBSUMES	SAME,SUBSUMED,SUBSUMES,ALTER,DISJOINT
ALTER	SUBSUMED,ALTER,DISJOINT
DISJOINT	DISJOINT

$R_{prev} = \texttt{SUBSUMES}$

R_{self}	Allowed values for R_{other}
SUBSUMED	SAME,SUBSUMED,SUBSUMES,ALTER
SUBSUMES	SUBSUMES
ALTER	SUBSUMES,ALTER
DISJOINT	SUBSUMES,ALTER,DISJOINT

$R_{prev} = \texttt{ALTER}$

R_{self}	Allowed values for R_{other}
SUBSUMED	SUBSUMED,ALTER
SUBSUMES	SUBSUMES,ALTER,DISJOINT
ALTER	SAME,SUBSUMED,SUBSUMES,ALTER,DISJOINT
DISJOINT	SUBSUMES,ALTER,DISJOINT

$R_{prev} = \texttt{DISJOINT}$

R_{self}	Allowed values for R_{other}
SUBSUMED	SUBSUMED,ALTER,DISJOINT
SUBSUMES	DISJOINT
ALTER	SUBSUMED,ALTER,DISJOINT
DISJOINT	SAME,SUBSUMED,SUBSUMES,ALTER,DISJOINT

Figure 6.14: Possible values for musical action signatures

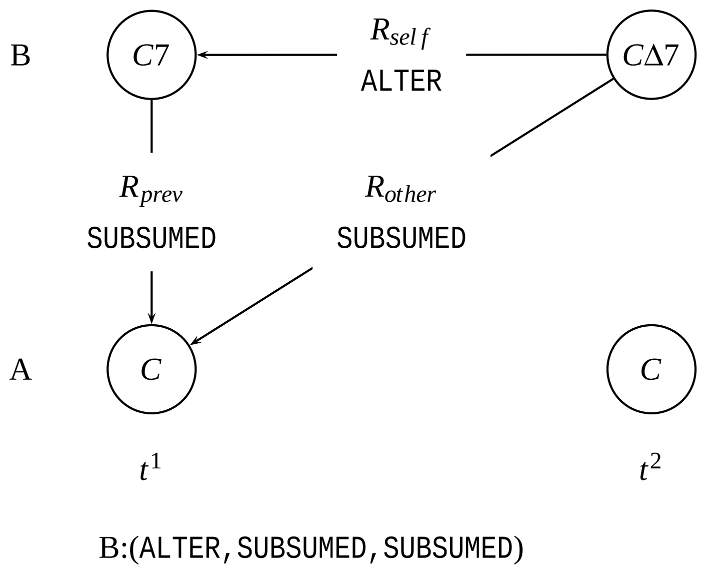


Figure 6.15: Example of an action signature being extracted from playing

State	A	B	Action Signature
s_0	C	⊙	A:(SUBSUMED,SUBSUMED,SAME)
s_1	C	C7	B:(SUBSUMED,SUBSUMED,SUBSUMES)
s_2	C	CΔ7	B:(ALTER,SUBSUMED,SUBSUMED)
s_3	D	CΔ7	A:(DISJOINT,DISJOINT,SUBSUMES)
s_4	D	D	B:(DISJOINT,SAME,DISJOINT)

Table 6.2: Example of representing a musical interaction as a series of actions

At this point in the piece, A may make a decision about what to play next. Assuming that A is using musical actions as a basis for playing music, there are two high level components to this decision:

- what musical action signature to embody.
- what high level feature to use to embody it.

Firstly, A must decide on an action signature to produce. This can be seen as choosing values for $R_{self}, R_{other}, R_{prev}$, or action signatures can be seen as discrete entities with some kind of performative *meaning* as discussed later in this chapter. The only constraint on this choice is that R_{prev} is already determined — it is the value between A's current playing and B's new playing, in this case `SUBSUMES`[5].

Suppose now that A decides to do something radical, and play some music which does not match what is currently happening — that is to say, a new value which is `DISJOINT` from both A and B's current playing. A would then decide to use an action signature of A:(`DISJOINT,DISJOINT,SUBSUMES`) for the next piece of output.

Once A has chosen this action signature, it is necessary to find some musical values which embody it. The choice now is out of all the values which satisfy the R_{self} and R_{other} relationships — in this case, a value must be found which is `DISJOINT` both from C and CΔ7. If A were to choose to play a D chord, the interaction would be as shown in Figure 6.16.

6.5.6 Extension to multiple agents

In a more general case, it is necessary to look at the actions which occur between several musicians at once. An easy solution for this would be to suggest that the interactions between each pair of agents be analysed separately. This has the benefit of not introducing any more theoretical complexity. However, there is then the downside of introducing a large number of interactions, which scales with $O(n^2)$. This also misses the fact than in most musical situations, the group will be in agreement on the majority of features.

To rectify this, the notion of *common acceptance* is introduced, based on the *musical common ground* which was introduced earlier. This is designed to capture the set of musical values which the group agrees on through their playing.

[5]To be clear, R_{prev} for this signature is between the same two values as R_{other} from the previous signature, but the direction of the relationship is opposite, since A is now the one carrying out an action.

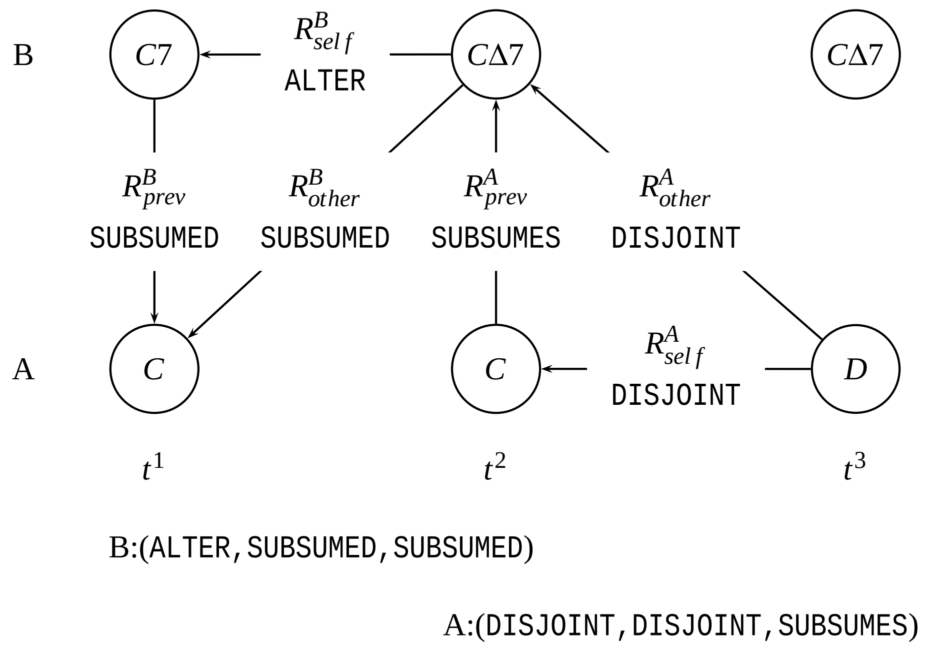


Figure 6.16: Example of two action signatures extracted from the playing of agents

Recall that there is a common ground of values which is the set of all values for each facet that an agent reasonably expects the others to understand. Looking at a single facet from this common ground gives the set of values which the agents are producing. Taking the *meet* of this set of values gives the most specific value for that particular facet which contains the playing of every agent - this is the *commonly accepted* value for that facet.

This formulation allows for an individual agent to construct groups of agents whose playing is in some way similar, and construct responses to the group, rather than each agent individually.

6.5.7 Reasonableness of $A_{context}$ and A_{state}

This section of analysis relies on three assumptions:

$A_{context}$: everything which is of interest to an agent about the musical interaction is contained in the context which it maintains.

A_{action}: all free musical actions consist of a change of state.

A_{state}: every change of state constitutes a musical action on the part of the agent whose value changed the state.

There is a question as to how reasonable these assumptions are, and whether they hold for all musical situations. Firstly, $A_{context}$; within the bounds of this model, this is a safe assumption — agents are modelled with a set of analysis routines for all of the features that they understand. Given this, it it is fair to say that everything the agent is capable of understanding is in the context. In a more general situation, it becomes somewhat tautological: the context *is* everything which the agent is interested in. Carrying on from this, A_{action} seems reasonable, as if the context represents everything an agent knows about the interaction, then anything which does not alter the context is invisible to the agent.

A_{state} needs more defence, however, on several counts:

- an agent may alter its values unintentionally — for instance, it may be attempting to play a constant rhythm, but be unable to keep a steady pulse, or it may play the wrong note in a chord.

- values may change in response to other processes. The major example of this would be if the agents are playing with some form of score, which dictates certain aspects of the musical surface. In this case, the agents would need a way to differentiate between intentional actions, and those which come about from following the referent, which are effectively conventionalised actions, as they are pre-arranged between the group.

- lastly, even if all changes constitute actions, do they all constitute *equal* actions?

Given these concerns, it appears that some measure of the importance or interest of an action would be useful, so that expected or trivial actions are noted as such, leaving more room to react to the truly novel and unexpected.

6.5.8 Relations to other kinds of actions

This chapter has been exclusively concerned with “free” musical actions, as opposed to conventionalised musical actions or extramusical actions. However, it would be useful to model the way in which other actions are present in the system, so simple descriptions are given here of how other types of action can be integrated into the current description. This is not a serious analysis of these different kinds of actions — each of which could be expanded on indefinitely — rather an illustration of a possible method for fitting them into this model of musical activity.

Conventional musical actions again start from the musical surface. The agent has some form of analysers, which are sensitive to certain patterns of musical events — the patterns which embody the conventionalised actions. These analysers will produce matches for their patterns in the output of the agents, creating a layer analogous to the featural layer from Section 6.2.2. However, it is not enough to simply identify patterns — it is necessary to examine them in context to see if a particular pattern is intended as an action by the agent which emitted it. An example:

- a group of agents are playing salsa music — traditional Cuban *Son*.
- the timbales player plays the pattern of strokes which can be interpreted as an *abanico*. An *abanico* is used to indicate the change from the slow beginning section of the piece to the faster middle section.
- the other agents hear these notes, and their pattern extractors notice that this is a potential *abanico*.
- the musical context is then examined — is this an appropriate place in the piece for an *abanico*? Could one be expected from the timbales player? If the correct conditions are met, then this pattern can be treated as an *abanico* action, and appropriate responses constructed. At other times, this meaning would not be attached to this set of notes — for instance in the middle of a drum solo — so no action would be created.

A similar story could be told about extramusical gestures — head movements are processed into events such as nods, which may or may not be processed into indications that it is your solo next. However, this would be overreaching the bounds of this work and will not be attempted.

6.6 Relations between Musical Acts and Musical Action Signatures

Chapter 5 gave a top-down description of a method for interpreting musical interactions in terms of intentional actions, which provided the impetus for creating this model of musical interaction. One of the prerequisites for using Musical Acts computationally is a formal semantics for the conditions of expressing a musical act, and constraints on the form in which it is expressed. In order to do this, an attempt will be made to generalise this to the performatives given in Section 5.1.3. The situation is taken to be: A is performing an action on facet f by emitting v_{new} after previously emitting v_{prev}; v_{old} is used to indicate the value played by the other agents — if differentiation is needed, then v_{all} refers to the set of individual values, and v_{common} refers to their common musical ground. The seven performatives can then be formalised as follows:

Propose requires that there was previously no accepted value for a particular facet, i.e. that $v_{old} = \odot$; hence, R_{self}, R_{other} must be `SUBSUMED` (since $\odot$ subsumes everything). Different formulations could be created around whether $v_{old} = \odot$ uses v_{all} and hence means that none of the agents have a value for f (Propose-New), or that v_{common} is used, so there is no common value for f (Propose-Discussion).

Confirm conveys an acceptance of an idea proposed by another; so, R_{other} should be `SAME` (although `SUBSUMED` might be allowed). It also requires that the new value was not previously contained in v_{prev}, so $R_{prev} \in$ `SUBSUMES,ALTER,DISJOINT`. Additionally, since $R_{other} =$ `SAME`, R_{self} must be the inverse of R_{prev}.

Reject indicates that the new playing is not accepted, so it must be different from it; hence $R_{other} \in$ `DISJOINT,ALTER`. It could also be argued that there should be no commonly accepted value for this facet, i.e. $v_{common} = \odot$.

Extend is the extension of currently accepted material, so R_{other} must be `SUBSUMED`. Since it is an extensions, R_{self} cannot be `SUBSUMES`, as that would indicate a withdrawing from the current position.

Alter involves altering a value which is already being used; so R_{other} must be `ALTER`.

Argue was an example of a composite act, where several people play without accepting each other's values; this could be modelled as a stream of acts where R_{other} was continually either `ALTER` or `DISJOINT`

Request is a conventionalised action, and so it cannot be modelled with action signatures — it is an expected response to a certain pattern of playing.

It can be seen that a particular musical action could be used in several different performatives; for example, (`SUBSUMED,SUBSUMED,SAME`) could be either Propose or Alter, depending on the musical context and the interpretation of any particular agent, which is as desired — attribution of intention should not be a formally derived process.

Performative	R_{self}	R_{other}	R_{prev}	Additional
Propose	SUBSUMED	SUBSUMED	any	$v_{old} = \odot$
Confirm	R_{prev}^{-1}	SAME	SUBSUMES ALTER DISJOINT	
Reject		DISJOINT ALTER		
Extend	$\neg$SUBSUMES	SUBSUMED		
Alter		ALTER		

Table 6.3: Formulation of example set of performatives using action signatures

6.7 Discussion and further work

There are several points which come up from this formulation:

- currently, all the different facets of analysis are treated separately. Real values in musical playing are unlikely to be entirely separate, so it would be useful to have some notion of an action which affected several values at once, especially since a single piece of playing may do this. A composite musical action could then involve the attachment of a set of musical action signatures to a certain part of the musical surface.

- the idea of repetition is not fully handled at the moment; in general terms, playing the same phrase repeatedly would generate actions on the first repetition, but then no new information would be added. However, this is not entirely in keeping with accounts of listening to music, especially where minimalist (and other repetition influenced) traditions are considered. A possible answer to this is that an agent's analysers could react to this repetition — as there are less large scale changes, some "lower level" analysers could become more sensitised, and the small differences between repetitions become more significant. This becomes more of an architectural decision for a particular implementation, but there is the possibility of giving different facets different levels of importance, to model the fact that certain changes are less obvious in the context of others; this would allow for these low level analysers whose output is generally ignored until such time as the large scale changes diminish.

- nothing is said about the roles which agents take in interactions — soloist/backing, teacher/student etc. This is intentional — it is a layer to be built on top of these structures. From the study of human interaction, e.g. Pelz-Sherman [1998], roles can be abstracted and defined in terms of the type of musical action signature which that role is likely to produce.

- in terms of the computability of objects in this domain, there are two procedures to consider: extraction of values for facets, and computation of the relations between two values. The second of these depends on the structure of the values used, but in general can draw on existing work in concept lattices and description logics to ensure that all values have easily computable relations. The extraction of values is less well defined, since values can be any aspect of music which can be analysed. This then becomes an implementation question of finding features with appropriate computational characteristics.

- all of this analysis has worked from the idea of a musical surface, where musical output is divided into perceptually distinct musical objects. However, this is not a necessary condition, rather it is a simplification to make it easier to discuss the formulation. Any representation of music may be used so long as there are analysers which produce lattice values from it, and it may be passed between agents in discrete fragments. There is also no reason not to use different analysers which work on different levels of representation; for example, the acoustic waveform of a performer's output could be used to derive timbral features, while a harmonic analyser worked on a representation involving pitches and durations.

- the role of the listener has not been addressed in depth; it is assumed that a listener can use the same techniques as a performer, simply without the ability to join in. The playing of any agent can be analysed *from the point of view* of that agent, or at least relations between values constructed with that agent in the role of *self*.

- one part of the theory which needs more exploration is the assumption that every interesting musical property can be represented sensibly using concept lattices. In response to this, it should be noted that the use of these structures was inspired as a more general version of the reduction hypothesis in Lerdahl and Jackendoff [1983]: their analyses of time-span and prolongational reduction depend on any given set of pitch events being seen as an elaboration of a simpler structure — this is fully in keeping with the idea of subsumption used in the lattice structures here. Although there are defined limits to the applications of this technique, it at least provides an example of representing complicated musical structure using lattice-like concepts.

- the formalisation of performative actions opens up the possibilities for virtual musicians to respond to the intentional aspects of human playing; with a given set of performatives, there is a limited set of labels which can be applied to a musical action, and some process could then be used to allow an agent its own strategy for choosing one of these labels — for example by modelling the other musicians, or referring to a database of past interactions. The formalisation also allows and encourages different sets of performatives to be experimented with, to find whatever set is most appropriate for a given circumstance. Desirable qualities of performative sets might be formal completeness — any action can only have one performative label, or all possible actions have *some* performative label — or resonance with natural language usage.

6.7.1 Relation to previous work

The most similar formulation of actions taken in a musical context is given in Pelz-Sherman [1998], detailed in Section 3.1.4. This account talks about several types of *i-events*, where musical information is exchanged between two participants. It would be useful to to define these in terms of musical act signatures, as they would then provide a computational implementation of an already existing theory about the nature of musical interaction.

The *i-events* given, and some possible action signature translations are:

Imitation is where one feature from an agent's playing is adopted by another. In terms of musical actions, the important fact is that R_{other} is `SAME`. However, it is also important that the previous playing did not contain this feature, so R_{self} and R_{prev} must be `ALTER` or `DISJOINT`[6].

Question and Answer events consist of some form of response to a cue, but the response need not use any features of the cue. Some of these are stylised, and hence not detailed here, but the general definition would be that R_{other} is either `ALTER` or `DISJOINT`. At this point, it might be useful to look at formulations

[6]it should be noted that R_{self} must have the same value as R_{prev} if R_{other} is `SAME`

across multiple values, such that some of the R_{others} are the same, while some are `DISJOINT`.

Completion/Punctuation occurs when one agent initiates a “directed movement”, which can be predicted to complete at a certain time, and another agent complete this gesture. This can be modelled with R_{other} becoming `SAME`, on a facet which looks at these kinds of directed gestures.

Interruption involves one agent playing in an undirected manner, which is decisively responded to by another. This could be modelled as a specific case of R_{other} being `ALTER` or `DISJOINT` on a feature which tracks some sense of musical direction.

So, musical action signatures can be used to provide a computational formulation for these types of musical exchanges, conditional on there being the necessary analysers to produce “musical direction” values. This is a fairly reasonable constraint, as to be a high quality musical performer, an agent would need to have some ideas about the directions that individual agents and the performance as a whole are taking.

Also, Pelz-Sherman [1998, page 130] mentions the idea of agent systems — subgroups of musicians whose playing is very closely aligned. The notion presented here of overlapping regions of common ground within a group of agents gives a way to analyse this computationally, and allow software agents to join these human agent systems.

Finally, a similar distinction is made between free musical actions and conventionalised: *i-events* cannot be part of some prearranged schema, and must occur in the moment, just as musical actions.

6.8 Conclusions

In this chapter, a computational model has been presented which can:

- model a group of agents improvising music together.
- model their analysis of music in a non style dependant manner.
- model the agents reasoning about the beliefs and actions of others.
- be related to existing theories and natural language descriptions of musical events.

Chapter 7

Implementation

This chapter details the creation of a real system which embodies the architecture laid out in previous sections. It talks about some design decisions which have been made in order to make the system technically feasible, and how this has resulted in an implemented system.

7.1 Introduction

This chapter deals with the details of implementing the infrastructure of an agent system which conforms to the system architecture outlined in Chapter 4. In terms of the agent outline, the parts under discussion can be seen in Figure 7.1. In more detail, the topics under discussion are:

Realising music discusses the issues involved in working with a real time musical system, the different temporalities involved and the conversion of internal structures into music which can be listened to.

Representing music deals with the Music Representation for Agents (MRA) system, which represent music as static scores, as Java objects and as messages in the agent system.

Implementing Agents covers the mechanics of how agents respond to messages and looks at some different types of agents which have been created.

Visualisations and Interfaces details the basic interfaces available for humans to observe and influence the agents while they are playing, and the mechanisms by which particular agent setups may be run.

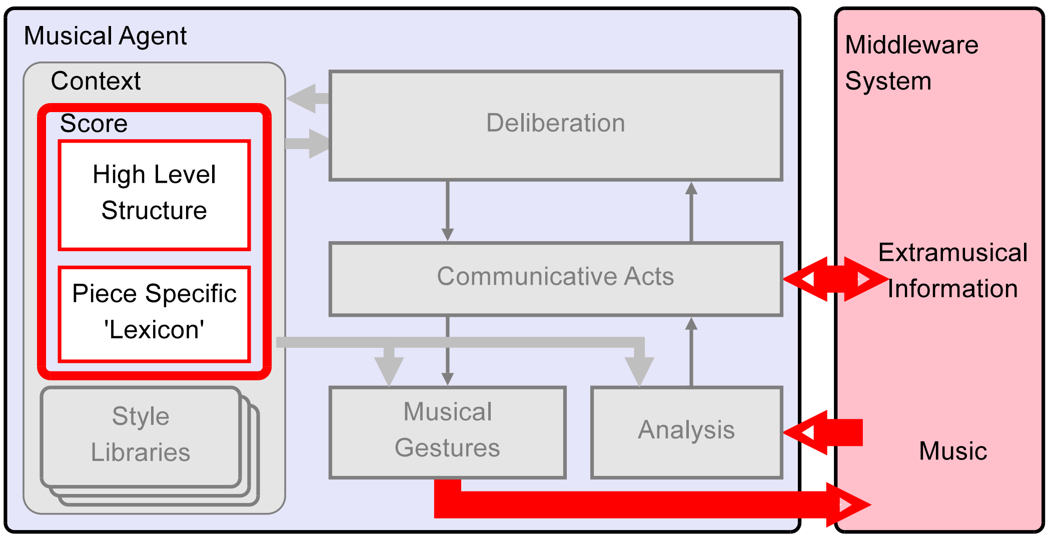


Figure 7.1: Parts of the agent overview covered by the Implementation chapter

The system currently consists of 264 classes, containing approximately 30K lines of code, with 65 high level test classes; Appendix A gives an overview.

7.1.1 Technologies

Java Like the rest of the project, the representation system is implemented using Java, a platform independent, bytecode compiled object oriented language [1].

JADE is an agent toolkit for Java, which provides facilities for creating and running agents, and handles message transport between agents in the system [2].

FIPA-SL is the "semantic language" developed by FIPA for agent communication, and is used by JADE for exchanging messages between agents [FIPA Specification].

Jade Ontologies are used by JADE to link concepts in a FIPA-SL message with Java objects.

MIDI is a standard for communicating with musical devices [MIDI Specification].

[1]http://java.sun.com
[2]http://jade.tilab.com

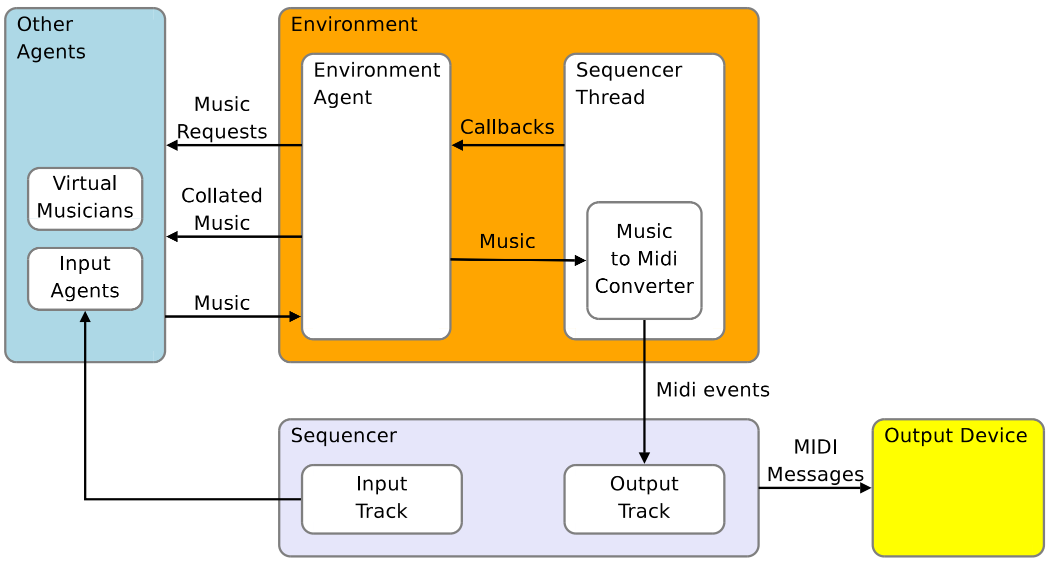


Figure 7.2: Overview of the music realisation process

7.2 Realising Music

This section looks at:

- the relations between musical time and physical time, and how this is used in the system.

- how the agent's representation of music is turned into sound.

- how the timing of the system as a whole is managed.

This all falls under the remit of the Environment and the Realiser (see Section 4.1.2). The overview of how this happens is shown in Figure 7.2.

The parts of this which this section details are:

- how the different types of time used in the system are reconciled.

- how the system manages internal timing.

- how music from the agents is turned into MIDI events.

The way that `RecordAgents` extract events from the `Sequencer` is touched on briefly in Section 7.4.4, and the loop which details the transfer of music between agents and the `Environment` is covered in Section 7.4.2.

7.2.1 Musical Time, Physical Time and Beat Tracking

The Musical Middleware specification does not constrain how timing is treated; the agents and the Environment may have whatever conception of time is appropriate for them. In the implementation of the system, however, it is necessary to make decisions about this. There are two types of time used by the system:

Physical Time is the standard system which is used in day to day timekeeping and scientific experiments. It is measured in seconds, which are constant units of time.

Musical Time is measured in beats and subdivisions. Events are related to an underlying *tactus*, which need not progress linearly with time.

Conversions can be made between the two types of time — for instance beat tracking and score following systems attempt to build a mapping from events in physical time to musical time, a music performance system may take music specified in musical time and transform it into sound events in physical time.

In order to keep the internal design of the system as simple as possible, all events within the system are stored in musical time, in terms of beats and fractions of beats. There are two conversions which then need to be made: events leaving the system must be converted to physical time, and events entering the system must be converted to musical time. There are many ways to perform this mapping, but in this case, the simplest possibility is chosen: a constant relationship is assumed between beats and seconds (or beats per minute, as it is more normally specified), and the system does not deviate from this.

While it is a limitation of the system that it must work to an isochronous pulse, it should be noted that:

- it would be possible to add a low level tempo tracking component at a later date; agents could be adjusted so that they took input in physical time, and converted it to musical time, with the higher level functionality remaining largely unchanged. There would be some additional capabilities such as intentionally altering the tempo, or responding to tempo alterations of others.

- When recording, many bands use a “click track”, which ensures that there is a constant beat throughout, which everyone follows. This makes it far easier for the recorded material to be edited later - if sections were at different speeds it would be more difficult to replace a section from one take with a section from another.

7.2.2 Converting MRA to MIDI

MIDI is a general standard for both representing music, and sending commands to a variety of devices which cause them to emit or shape sounds. MIDI *messages* can be used to start or end notes, set the instrument a particular device is playing, alter parameters of the sound which is being used and so on. A MIDI *event* is a tuple of a message and a time, and can be stored in a *track* for later playback by a *sequencer*, which will ensure that the events are emitted at the appropriate time. MRA `Fragments` are converted into MIDI events, with each note being converted into a *NOTE_ON* and a *NOTE_OFF* event. Each `Fragment` which the agent sends out contains information about the agent who played it; a name, its position and what instrument it is playing on.

The following features are taken into account:

- the timing of the events which start and stop the note are determined from the onset and duration parameters of the MRA note; Events in MIDI tracks are timed in "ticks", where a tick is some proportion of a beat[3], so the beat timings of MRA must be converted to ticks. This is still "musical time" — events can be played back by the sequencer at any speed. The transformation from MRA time to MIDI time is a linear transform, multiplying the floating point beat value of the MRA time by the number of ticks per beat to get an integer number of ticks.

- the volume and pitch of the note on message are determined by the volume and pitch of the MRA note.

- the agent's position within a virtual space alters certain characteristics of the output: the y-axis is used to affect the volume of the note, so that agents closer to the back of the AgentSpace (Section 7.5.3) are quieter; the x-axis is used to send a "pan" message, which shifts the output of that device towards the left or right of the stereo field.

- the instrument which the agent is playing is mapped to the GM instrument set, which defines a standard set of instruments present on most general purpose

[3]different midi formats provide for different numbers of ticks, or Pulses Per Quarter-note (PPQ)

MIDI devices. A "program change" event is generated when each agent starts playing to ensure that the correct sound is generated.

- the agent's identifier is used to make sure that each agent is assigned to a particular MIDI "channel". Most MIDI devices are multi-timbral, and can playback sounds with a range of voices independently. Having each agent on a separate channel means that pitch and timbre can be set individually.

7.2.3 Turning music into sound in time

The midi events which have been created must now be played. The Java platform provides a `Sequencer`[4], whose job it is is to turn events, specified in musical time, into MIDI messages which are output in physical time. The `Sequencer` has a set of tracks containing these events. In order for the Environment to produce sound, the events produced from MRA music are added to one of the sequencer's tracks. This is where the scheduling latency t_{sched} comes in — the sequencer is always reading from the `Track` ahead of where it is playing, so events added too close to the current time will not be played. t_{sched} is the minimum time which must be left between scheduling notes and their intended time of output to guarantee that they will play at the correct time.

7.2.4 Timings for the agent system

The `Sequencer` is currently the only bridge between musical and physical time, so it is used to force the timing constraints on the `Environment` agent. The `Environment` can set itself to listen for meta-events[5]. Every time a new fragment of music is added to the `Track` which the `Sequencer` is playing, two meta events are added:

REQUEST_MUSIC_MESSAGE prompts the `Environment` to request the next fragment from all currently playing agents.

FRAGMENT_DEADLINE_MESSAGE prompts the `Environment` to take whatever music it has received from the agents, schedule it, disseminate it and then schedule the next two meta events.

This allows the rest of the `Environment` to be written without worrying about real time issues - everything runs in musical time, and does not worry about scheduling.

[4]javax.sound.midi.Sequencer
[5]midi messages with specialised, non-standard meanings

Furthermore, all communication with the `Sequencer` happens in a dedicated thread, to minimise the possibility of events in the agent system causing timing deadlines to be missed.

7.3 Representing Music

This section looks at the way in which music representation is implemented in MAMA. Chapter 4 defined several properties of the musical representations to be used in this system:

- the division of representation into structural components and objects attached to the structure.
- the criteria that the system should be structured, addressable, flexible, non-specific, open, powerful, extensible and natural.
- the three forms which the music representation has to exist in
  - in the agent's memory,
  - on the network,
  - in files for creation and storage.
- that there must be a representation of fragments of music, representing the music played by each agent, that can be passed around the network.

It was also noted that there are three forms in which the music needed to be represented. These are (see Figure 7.3 for a lifecycle diagram):

Files are used to store music on disk; scores are created in the MRA language by composers, and later read into the MRA system.

Musical Objects are Java representations of the musical entities, which are manipulated by the agents to create more musical objects representing their output.

Network Messages are used to transport the output of all the agents around the system.

There are two parts to the representation system: the concepts which it embodies, and the ways in which these concepts are represented at different times. The discussion of the concepts and entities to be used can be found in Section 4.3.7, while the

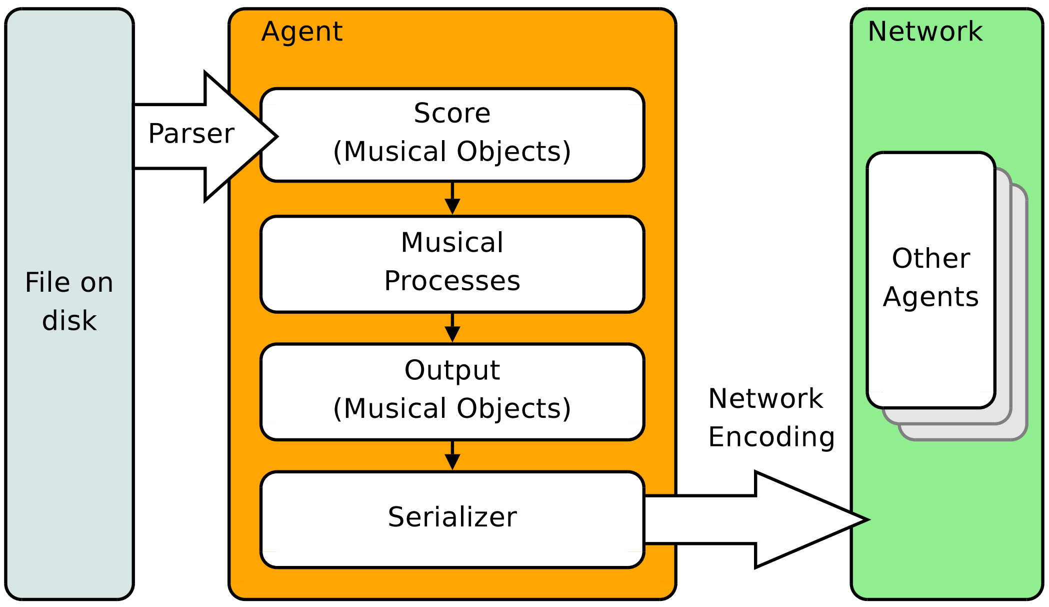


Figure 7.3: Lifecycle of music representation

remainder of this section deals with the representation of these concepts in different forms.

7.3.1 Language Syntax

In general terms, MRA is a hierarchical representation, consisting of objects which contain other objects. Certain types of object have names, and all objects may have attributes.

The MRA language draws inspiration from two sources: XML[6] and FIPA-SL; both of these are hierarchical languages, containing named elements with attributes attached. MRA is written from scratch, in order to make sure that it can be altered as necessary to fit the constraints; it is very syntactically close to FIPA-SL, and architecturally similar to XML, the exception being that MRA is domain specific, so some constraints on the ordering and nesting of objects are enforced at a language level.

The core syntax for MRA is relatively concise. Objects are delimited by parentheses, with the type of the object as the first string inside them, and the object's name as the second string if it is a named object:

```
( ObjectType name )
```

[6]http://www.w3.org/XML/

Attributes are written on separate lines, with the attribute name followed by a colon and the attribute value:

```
( ObjectType name
    Attribute1:     value1
    Attribute2:     value2
)
```

Objects are named in MRA to allow them to be addressed, both by agents and internally within the score - to allow reuse of similar objects where possible. In an MRA file, a named object may be either referenced — by writing just the object's type and name — or defined — some attributes or contained objects are included; a symbol table keeps track of all the named objects, and creates placeholders for objects which have been referenced but not defined. Objects which have been referenced but not defined, or defined more than once will produce warnings when the file is parsed.

For example, in the following code, three things are happening:

```
( Object a )
( Object b
    Attribute: value
)
( Object a
    Attribute: value
)
```

- Object a is referenced, but not defined.
- Object b is defined and used.
- Object a is defined and used.

7.3.1.1 Pieces and Sections

The fundamental unit of structure in MRA is a `Piece`, and an MRA file must contain exactly one `Piece`. `Pieces` contain some number of `Sections`, each of which may contain either more `Sections` or some number of `Channels`, as outlined in Figure 7.4. One of the top level sections *must* be called "main", and this is used as the starting point for the piece. This is in keeping with the specification from Section 4.13.

For example:

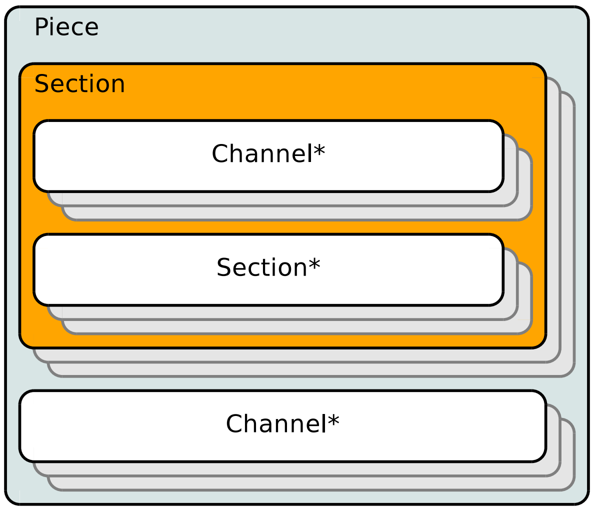


Figure 7.4: Object structure in MRA

```
( Piece myPiece
    Rhythm:     funk
    Tempo:      fast
    ( Section main
        ( Section intro )
        ( Section verse1 )
        ( Section verse2 )
        ( Section chorus )
    )
)
```

defines the beginning of an outline of a piece of music.

7.3.1.2 Channels, Spans and Notes

Channels are defined just like most other objects, with the word “Channel”, and a name, which is generally used to describe the type of events which the channel holds. For example:

```
( Channel Chords
    ( ChordSpan: C minor, 4.0, 6.0 )
)
```

would be used to define part of a chord structure — that a C minor chord was being used between beats four and six of the section the span was contained in. This also illustrates a small piece of syntactic sugar: since the events in channels are often simply specified but numerous, they can be specified without naming the attributes, using an implicit ordering (defined by the particular type). For example, rather than having to write all of the attributes of a note individually, i.e.

```
( Note
    Onset:      1.0
    Duration:   2.0
    Pitch:      C4
)
```

the attributes may be passed as a tuple:

```
( Note: E, 1.0, 1.0 )
```

or for a gracenote (where no duration is specified):

```
( Gracenote: E, 1.0 )
```

Certain channel names will be recognised by the agents; in general a channel called “Notes” will indicate that it contains notes the agents should consider playing; if an agent knows it is playing a particular part, it will look for a channel with that name containing the notes to play.

7.3.2 Music in Memory

When an agent has read in a score, it needs programmatic access to it; this should be fast, and allow the agent to perform any necessary operations as succinctly as possible. This is provided by a hierarchy of Java objects, in the package `com.mo-seph.mra`. There are two main families of objects: those descended from `Section`, which are structural, and contain other structural units, and those descended from `Channel`, which contain objects with timing information. A `Fragment` is a specialisation of `Channel`, dedicated to note-based data. These match the structural description of the language given in Section 4.3.7.

For working with music-as-played, there are some additions which go outside the basic MRA language. Firstly, `Fragments` are associated with an agent which has played that fragment. Secondly, `Fragments` may be combined into a `Score` (a slightly inappropriate name, which remains for historic reasons), which represents the output of a whole group of agents for a certain time period, and can be indexed by agent ID.

7.3.3 Music on the Network

Music must be transmitted round the network for other agents to “hear”. Ideally, this should be compact, fast to decode, and understandable by agents independently of the technology platform used.

At present, there are two methods used, which both have advantages and disadvantages:

Serialised Java Objects are fast and easy to encode and decode, as they are a copy of the representation in memory of the objects the agents are already using. The downside is one of compatibility — this cannot be guaranteed across revisions of Java, and it is definitely not possible to interoperate with other languages.

FIPA-SL is FIPA’s standard agent communication language, and is human readable. It is a platform independent way to encode the content of messages, and allows for interoperability. However, it can be slow, and there is overhead involved in creating the ontologies necessary to support its use.

Currently, the time-critical, high throughput music messages are implemented using serialised Java objects, although an experimental SL based mechanism has been used for some other messages — the rationale for this is to allow development to progress easily, but leave the door open for creating a standards compliant version later on.

7.3.3.1 Using FIPA-SL for message content

In order to use SL messages with an agent system, we must provide an ontology, specifying what terms are allowed in the domain, and what values they may take; so long as an agent understands both the ontology and the content language, it may participate. In order for a message to be sent and processed in JADE, the following steps are taken:

- The sender starts with some data structure, representing the message to be sent.
- The datastructure is encoded according to the agent language used (in this case FIPA-SL).
- The encoded string is put into a message, and sent to the receiver.

- The receiver parses the message according to the syntax of the communication language, resulting in an abstract data structure.
- Using the ontology, the receiver can convert this abstract data structure into appropriate objects.

Implementing this on another system should (in theory) be a case of loading the ontology, and making sure there is an appropriate set of objects to use with it.

7.4 Implementing Agents

The system design created a simple contract between the Environment agent and every music producing agent in the system, shown in Figure 4.12. This can be summarised as:

- the Environment will ask for musical output at the appropriate times, i.e. just in time to schedule it for playback.
- the Environment will disseminate collated agent output as quickly as possible.
- each agent must respond with a reply immediately when it is asked for output.

Since the design of the environment has already been discussed, this section covers the main musical loop, in terms of the messages which are passed between agents, and the implementation of music producing agent's responses to events in this loop.

7.4.1 Concurrency in JADE

The JADE system handles concurrency as follows: [Jade Guide, page 24]

- every agent runs in its own thread.
- each agent has a set of `Behaviours`, which are scheduled in a round-robin, non-preemptive fashion.

Behaviours are used to create the agent's responses to messages: when a message is received, the scheduler attempts to match it to the behaviours active in the agent at that time. Once a behaviour is found, it begins execution, and carries on until it has finished. Since the scheduler cannot interrupt execution, it is up to the programmer to make sure that behaviours complete in a reasonable amount of time, or to split them into smaller chunks which can be executed one after the other.

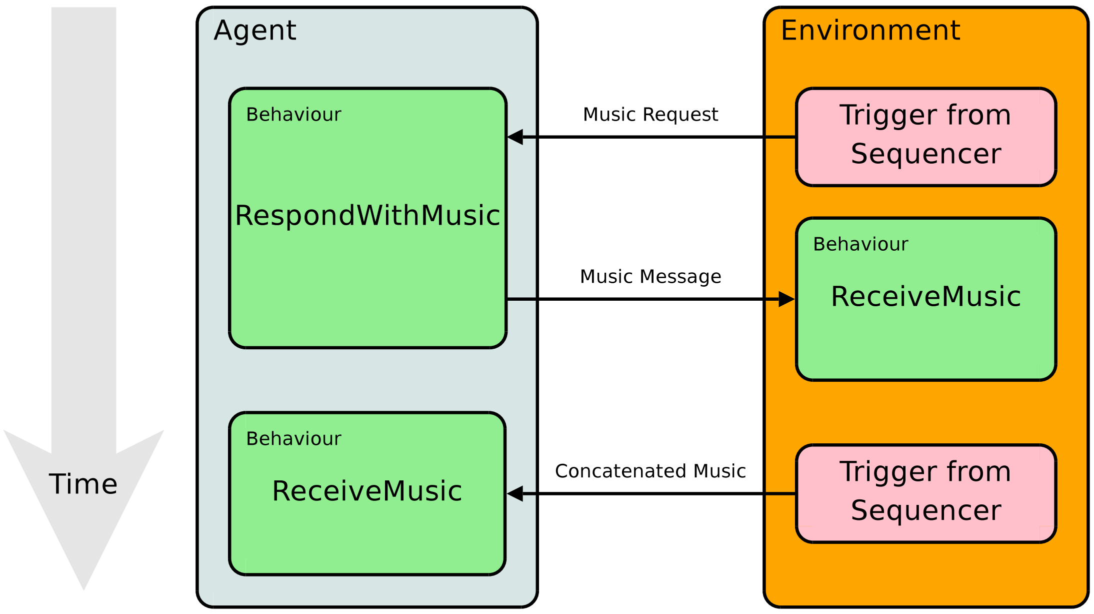


Figure 7.5: Message flow in the Main Musical Loop

7.4.2 Main Musical Loop

In the main musical loop of the agent system, there are only three `Behaviours` used: (see Figure 7.5)

`MusicCollection` in the `Environment` constantly waits for music messages to be sent from the rest of the agent system. When these messages are received, they are added to the current set of received music. The rest of the `Conductor`'s operation happens in response to internal events (generated by the sequencer), so no `Behaviours` are used.

`RespondWithMusic` in `MusicalAgent` fulfils the agent's main contract - it immediately returns the next chunk from the agent's output buffer.

`ReceiveMusic` in `MusicalAgent` is triggered by the `Environment` disseminating the output of all the agents for the previous fragment.

7.4.3 Analysing and Creating Music in Agents

In order for the `RespondWithMusic` behaviour to function correctly, the agent must have music already prepared in its output buffer. Since the agent does not have any

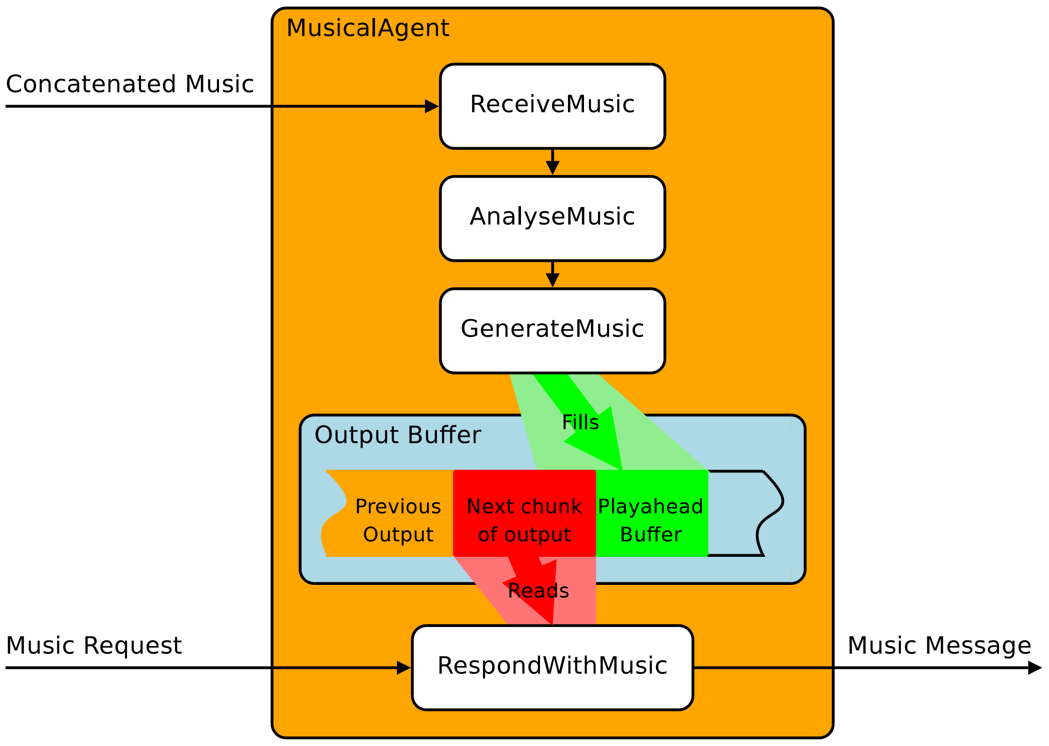


Figure 7.6: Musical Agent generation of music in response to messages

setup for realtime operation, the creation of this music must be triggered by an external event. There are only two possibilities here - the request for musical output, and the delivery of output from the other agents. In order to allow the maximum amount of time for music generation after music is received from the other agents, this is used as the trigger for both analysing their output and generating new music - see Figure 7.6.

The sequence is as follows:

- the agent receives a message containing a `Score`, representing the musical output of all the agents for the previous timeslice.
- the agent sends this to its analysis system for analysis and storage.
- the agent generates some music. It generates music until the output buffer is filled up to the end of the next timeslice, plus a “playahead buffer”. This extra buffer can be used to ensure that the agent has output ready, even if the current generation stage has not yet finished — at the expense of increasing the time it takes the agent to respond.

This model for behaviours is not strictly in keeping with the JADE philosophy - the agent is doing a lot of work in a single behaviour, without allowing control to be passed to other behaviours. It would be more architecturally sound to have this behaviour execute in its own thread; however, at this stage, the extra work which would be needed to ensure correct concurrency operations and the overhead of doubling the number of threads in the system is not considered worthwhile.

7.4.4 Bestiary of agents

This section describes some of the agents which have been created to test the system and to add necessary functionality.

Firstly, there are two classes which provide starting points for the development of useful agents:

MusicallyAwareAgent is the base class for all agents which have anything to do with music. It contains methods for serialising and deserialising music from/to messages, finding other agents, creating musical behaviours and interacting with the agent environment.

MusicalAgent provides all the facilities an agent needs to play music except actually generating the notes; it maintains an output buffer, and calls methods to fill it at appropriate times, handles all the messages an agent must respond to, and has a `Location` and an `Instrument` for producing sound.

Next, a set of agents which play music in some way, all derived from `MusicalAgent`:

ScoreAgent is the agent used for most of the situations discussed; it plays some kind of score, by rendering the notes in response to the playing of other agents. This agent contains the analysis, generation and reasoning systems discussed in the rest of this thesis.

CopycatAgent is a simple agent for demonstrating the workings of the agent system. Each copycat will listen to the output of another agent, and repeat it after a certain delay. When that agent stops playing, the copycat will find another agent to listen to and copy. Playing with a large group of copycats creates a repetitive texture which responds to human input.

Finally, some agents which deal with input, output and visualisation:

RecordAgent provides a way for humans to play music with the agent system. Each `RecordAgent` is tied to a particular MIDI device and channel; it uses the same `Sequencer` which provides timing for the `Environment` as a way to record notes and transform them into MRA representations.

PlaybackAgent allows the playback of prerecorded MIDI files. It progressively reads through a given track in a given MIDI file, and copies the notes into its output buffer.

OSCAgent allows communication with the agent system via OSC. It allows external systems to send messages to individual agents.

SpaceAgent provides a graphical interface to the agent system. It allows the location and status of agents to be altered, and agents to be created or destroyed.

7.5 Visualisations and Interfaces

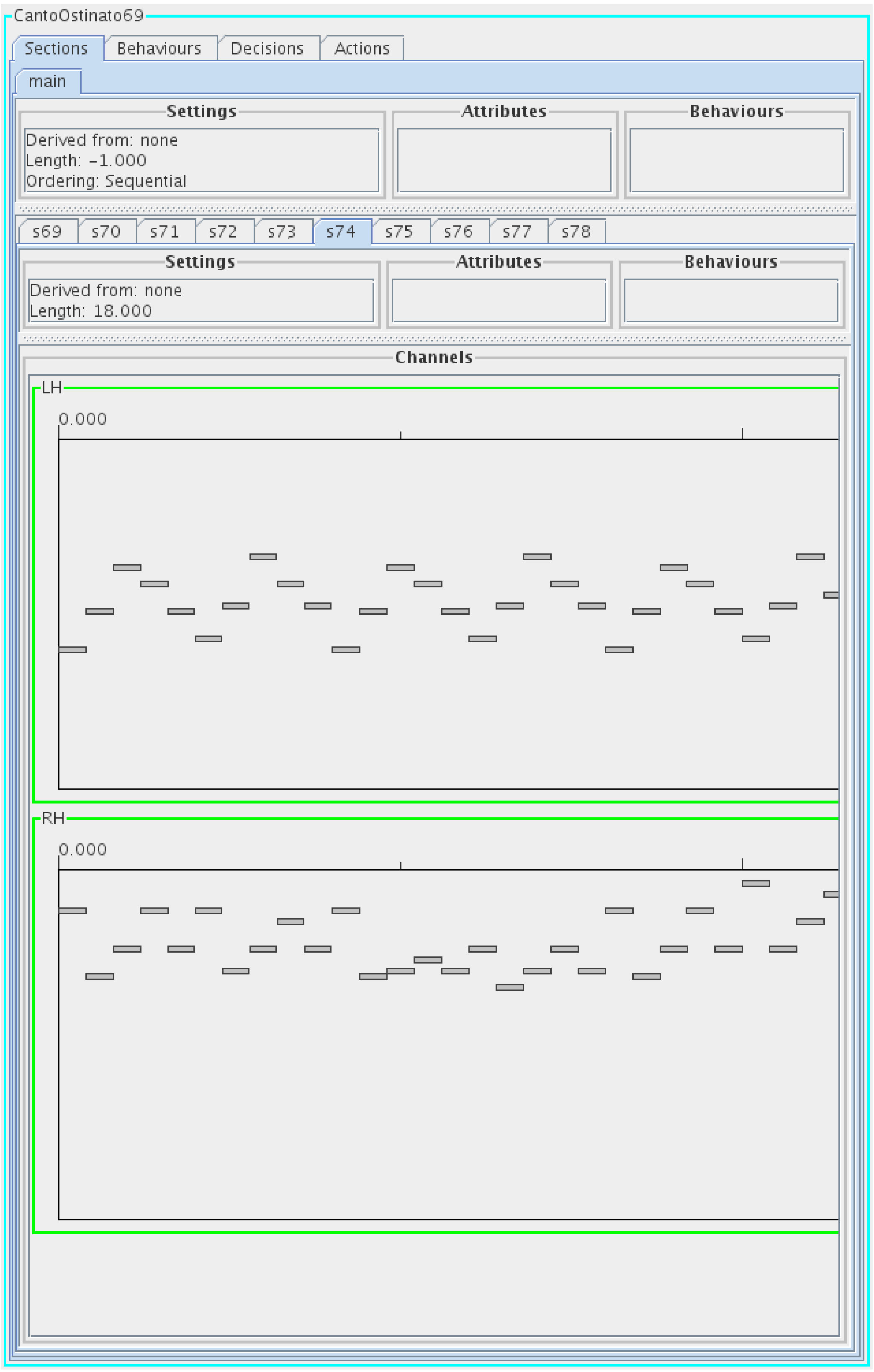
7.5.1 Interfaces

At this stage, the interface to the agents is relatively basic. The agents are started by a dedicated `AgentRunner` class. This class creates:

- a Conductor agent.
- a set of musical agents. Each agent is given a name, and the name of an instrument to play. Additional configuration (e.g. which type of Reasoner to run) can be added here.
- a GUI agent to allow the user to visualise and control the performance.

7.5.2 MRA

To aid the development of pieces in MRA, a set of visualisation tools are provided. At present, these are read-only, and display the structure of the piece in a given file, but this could be expanded to add full editing capabilities. An example is shown in Figure 7.7



Derived from: none
Length: -1.000
Ordering: Sequential

Derived from: none
Length: 18.000

Figure 7.7: Visualisation of part of the score of Canto Ostinato using the MRA visualisation tools

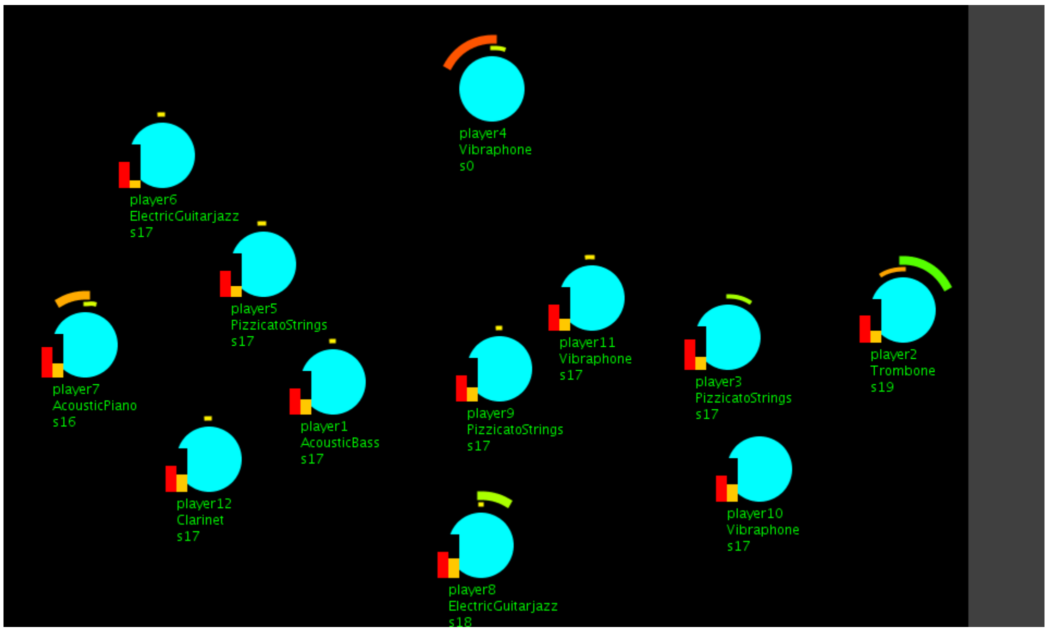


Figure 7.8: AgentSpace: overview

7.5.3 Agent Space

The AgentSpace graphical interface gives a range of feedback to the user; Each agent is represented by a circle, situated in a virtual space, as shown in Figure 7.8. The display represents information about the agent as follows (see also Figure 7.9):

- the circle's position represents the agent's position in space.
- a text annotation details the agent's name, instrument and what section it is currently playing.
- the circle's colour can be used to represent different things; in one case study it is used to give an indication of how far through the piece the agent is (see Section 10), while in another it relates the agent to a physical counter used to control it (see Section 10.2.3).
- two bars at the side of the agent represent density (the proportion of the time the agent is playing notes) and dynamics (the average velocity of the notes the agent is playing) to give the user some idea of the different output of the agents.

The only mode of interaction with this view is the mouse. The user can pick up agents and move them around the room — or more correctly, the user can *request* the

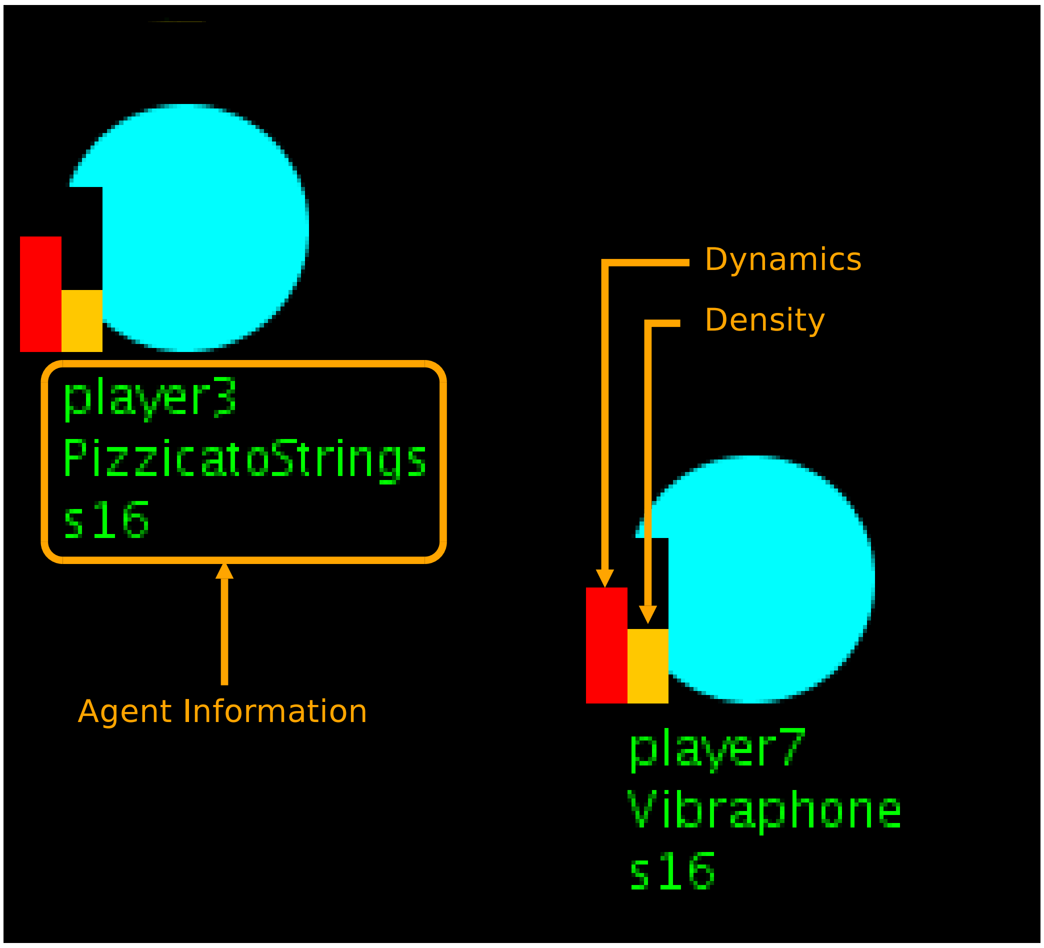


Figure 7.9: AgentSpace: detail

agents to move around the room; in the end, the agent has the decision over whether to move or not. Each agent has a context menu (right click) which allows the user to:

- "kill" the agent, so it leaves the current performance permanently.
- ask the agent to take a break, or return from a break; this makes the agent stop playing as soon as possible, or resume playing as soon as possible respectively.
- send a "bump" message; this is a contentless message, which agents may be set up to respond to in different ways; Section 10 has more information on uses for this.

There is also a general context menu, which allows for the creation of new agents, and also for all agents to be sent on or recalled from a break.

7.5.4 Agent Interfaces

Each agent has its own GUI, which can display a range of things, depending on the agent and the launch parameters. At present, there are two main views used:

Analysers shows the features extracted from the playing of other agents by this agent's analysers. (see Figure 7.10)

Reasoner shows the output of this agent's deliberative process (see Figure 7.11), in the form of its `RenderPlan` (see Section 8.2.3).

Both of these views are purely for visualisation — they do not allow a user to interact with the agents.

7.6 Discussion

This Chapter has dealt with the construction of the infrastructure of the system, so discussion of the possibilities once this infrastructure is available are found elsewhere, in Chapter 10.

7.6.1 Performance and Timing

One of the first questions to ask of a musical system infrastructure is how good is it at producing music? In this case, the main issue is timing — sound is provided by MIDI synthesisers, and the notes are provided by whatever kind of agent is used. The system

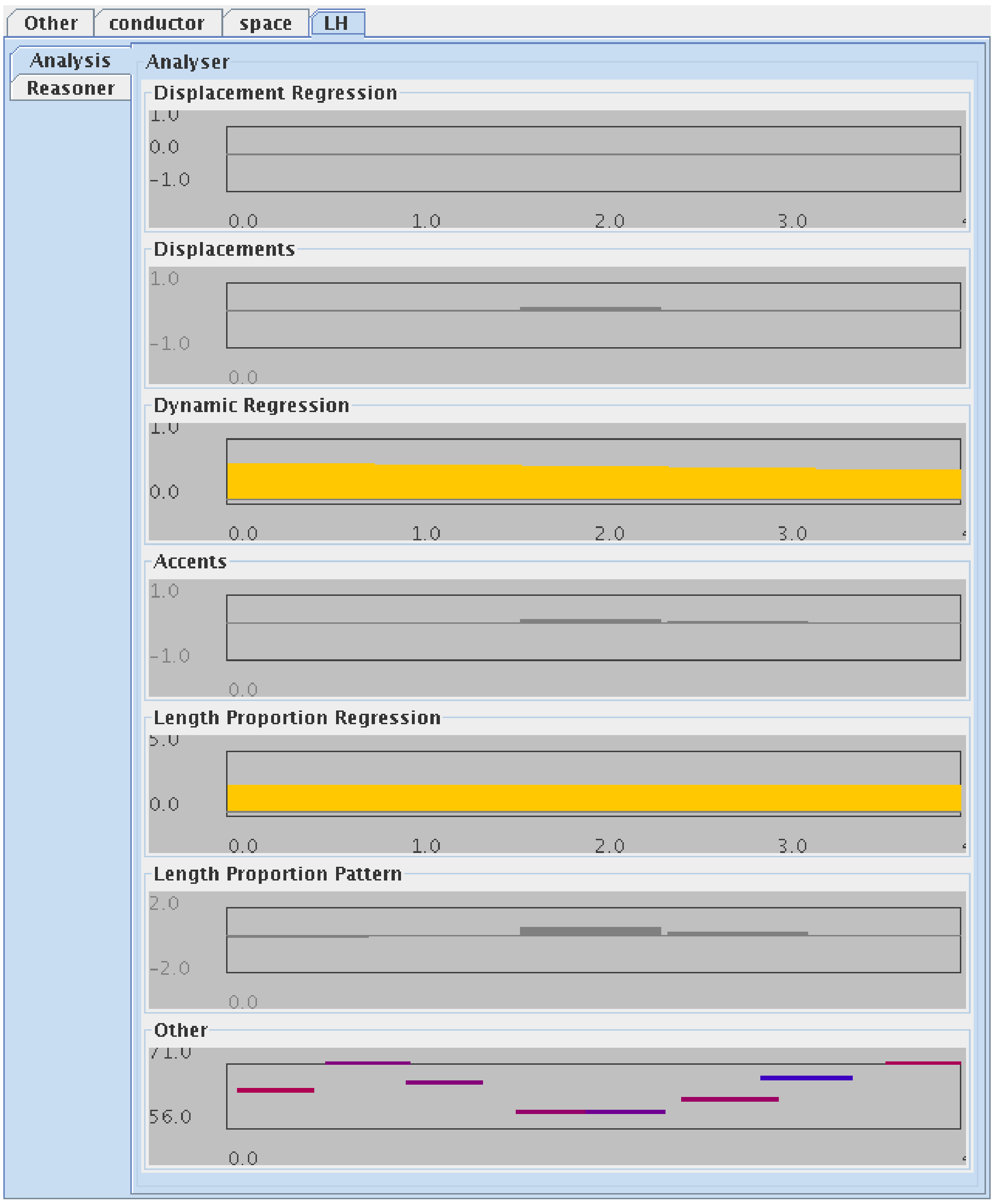


Figure 7.10: Agent analyser screenshot

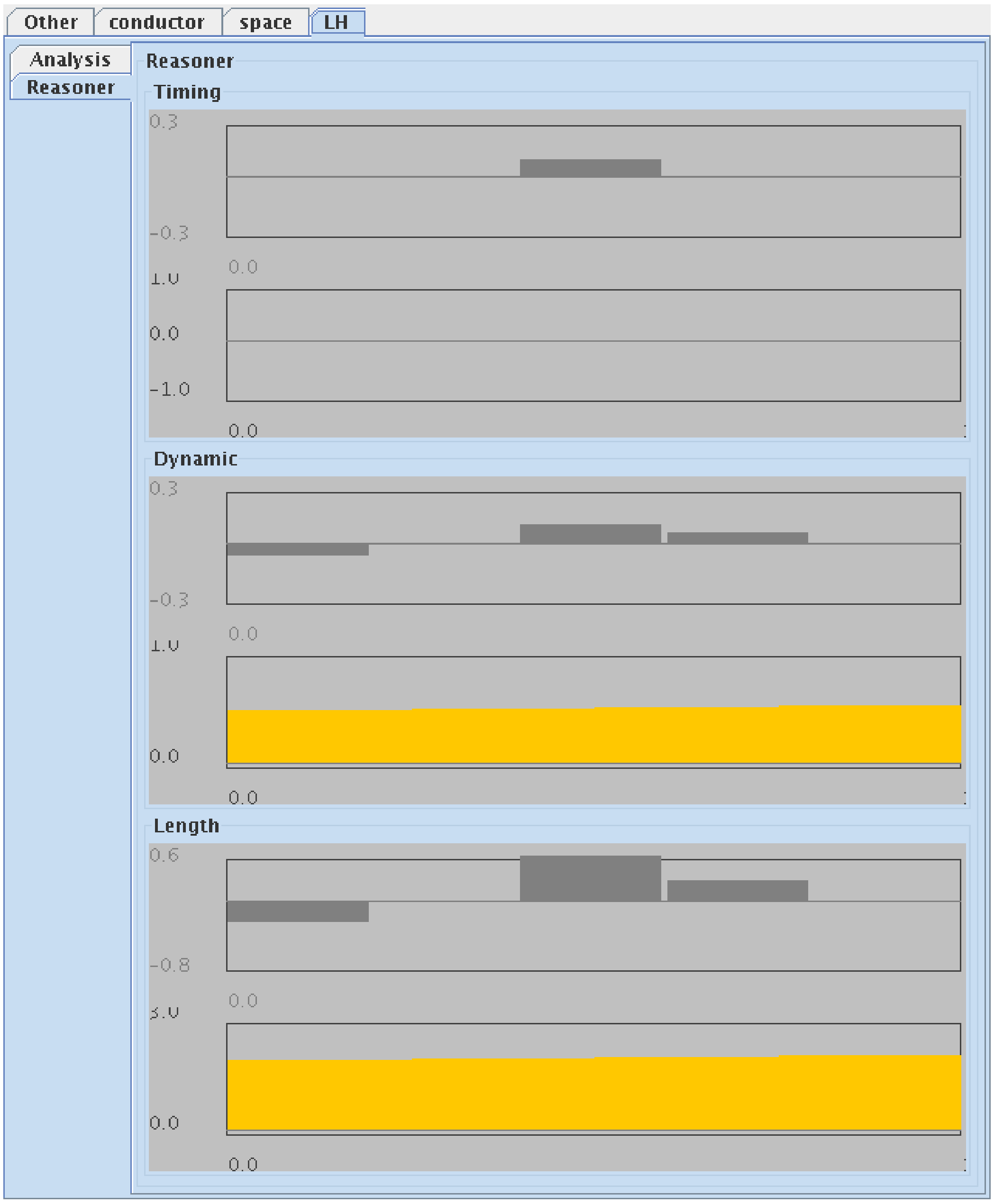


Figure 7.11: Agent reasoner screenshot

mx512M
ms512M
X:+UseParallelGC
X:MacGCPauseMilis=1
X:+UseAdaptiveSizePolicy
X:GCTimeRatio=9999

Table 7.1: Arguments to the JVM for optimal musical performance

relies on the Java sequencer classes to provide timing, and there are some issues which have been worked around here. One of the dangers of using Java for realtime tasks is that it is a garbage collected language, and the programmer has relatively little control over when and how garbage collection is performed. In a default Java VM, memory will be used up to a certain level, at which point the GC will run, and all other operation will cease until it has finished. This is clearly not acceptable for musical operation, as it introduces pauses, generally around 200ms, but sometimes much longer into the output of the system. Recent versions of Java (v1.5+) provide some settings for GC *ergonomics* — ways to modify the behaviour of the GC to suit particular tasks. These have been set as shown in Table 7.1, which provides relatively solid operation.

Another question which could be asked is how many agents can the system run, and what is the scalability like? At present, approximately 20 agents can perform In C together, on a Pentium Mobile 1.7GHz laptop. These agents are relatively simple (see Section 10 for details), but they still perform some level of musical analysis, and producing notes. The fact that there is a central Environment agent may seem like an issue with regard to scalability, as it provides a single point of concentration for musical messages. However, the tasks of the Environment are relatively easy — it need only exchange music and sonify it. Hence, as the complexity of the agents grows, it would be possible to use more and more remote machines to run agents, while keeping the Environment on a powerful local computer so the output can be heard.

7.7 Future Work

There are some limitations in the way the system currently functions, which could be addressed by future work:

- the fact that the system works to an isochronous pulse is a deliberate simplification — or at least the fact that this pulse is treated as a musical object. It would be desirable to allow the agents to keep their own notion of time and tactus, or an intermediate simplification would allow the Environment to match its tactus to human musicians, and all of the individual agents conform to this beat.
- at present, only humans playing MIDI instruments can interact with the system. The fact that the agents can take a higher level view of music than the note level (Chapter 8) opens up possibilities for extracting features of acoustic instruments to use as input, without having to accurately transcribe each note.
- the fragment size is currently the same throughout the system, and set in a configuration file. This could be changed to a dynamic property, managed by the Environment, and could be altered on a per-agent basis, so that agents with higher latency have to work at a larger fragment size, or more fragments behind. A whole investigation of the possibilities of distributed agents is possible, which would become more interesting as more computationally intensive agents were produced.
- a wider range of behaviours should be specifiable through the score; in order to implement the group-following behaviour for In C, it was necessary to use a custom module; it would hence be interesting to collect a set of different behaviours specified in musical scores from different canons and construct an extension to the representation language which was capable of representing them concisely and intuitively.

7.8 Conclusion

This Chapter has presented an implementation of the Musical Middleware specification, which provides an environment for musical agents to run, exchange music, read human generated scores and interact with humans musically and through a GUI. This demonstrates that the middleware specification is implementable and relevant to creating musical agents.

Chapter 8

Analysis and Generation in MAMA

This chapter describes the mechanism by which music is analysed and generated in the MAMA system.

8.1 Introduction

Figure 8.1 graphically indicates the parts of the agent architecture whose implementation is discussed in this chapter; specifically, the processes detailed here are:

- extracting high level features from music using a variable set of `Features`.
- using RenderPlans, which combine features with notes from the score, to produce finished output.
- a simple system which uses features extracted from the input to produce output.
- maintenance of a context of the features extracted from the playing of the other agents.

The intention here is to give an explanation which is low-level enough to talk about the issues involved in implementing this system without becoming swamped in implementation details. It should also be noted that the musical features being discussed here are not of particular interest for their own sake; the aim of this section is to discuss the architecture within which these features are extracted and used, with the implicit assumption that the current feature set could be expanded with more complex and interesting features. This use of a "microworld" with a limited feature set aids clarity when experimenting with complex, realtime systems by reducing the number of factors which need to be considered at any given time.

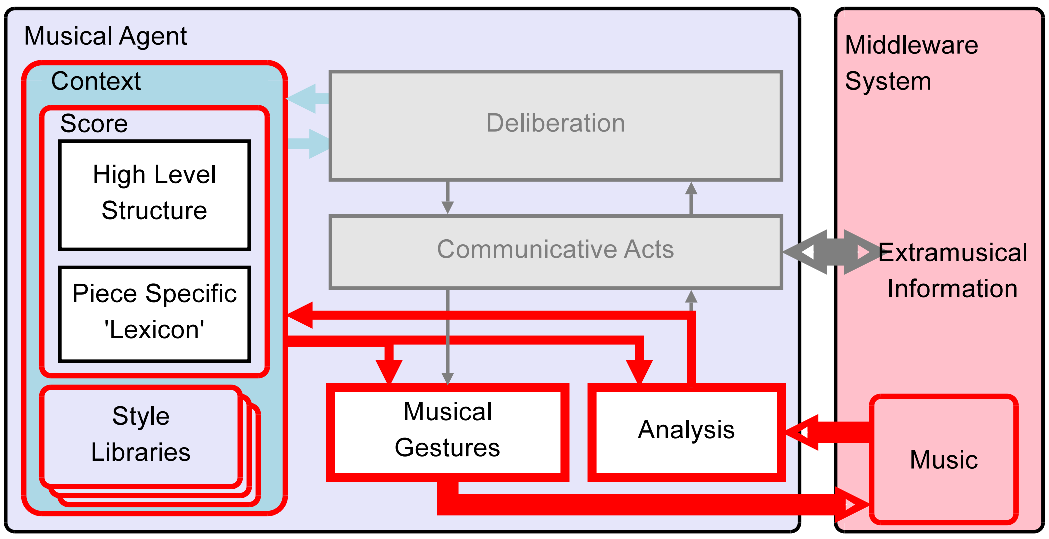


Figure 8.1: Parts of the agent system covered in Analysis and Generation Chapter

Throughout this section, there are two notions of notes, or note objects which will be used; played notes, and "ideal" (or scored) notes. It should be noted that this is not an indication that there is a "correct" or "perfect" time to play notes, rather that this type of musical analysis is enabled by imagining a metrical grid on which scored notes are placed, and relating the notes which are played to this grid. On a related note, although the system at present is constrained to a strict isochronous pulse, this is an artifact of implementation; ideally another process would perform beat and tempo tracking (probably in a feedback loop with the analysis section), and produce the notes pinned to a metrical grid which are used here.

8.2 Analysis

The design of the analysis architecture is based on the model of musical interaction set out in Chapter 6, particularly Section 6.2.2. Here, an agent's understanding of the musical surface is built using any number of descriptors, with the constraint that the values taken by these descriptors must be situated on a semilattice. So, the guiding principles for implementing the analysis system are:

- the set of features and types of analysis must be modular and expandable, to allow for different styles of music and expansion of agent capabilities.

- the values of these features must lie on a semilattice for that particular feature.

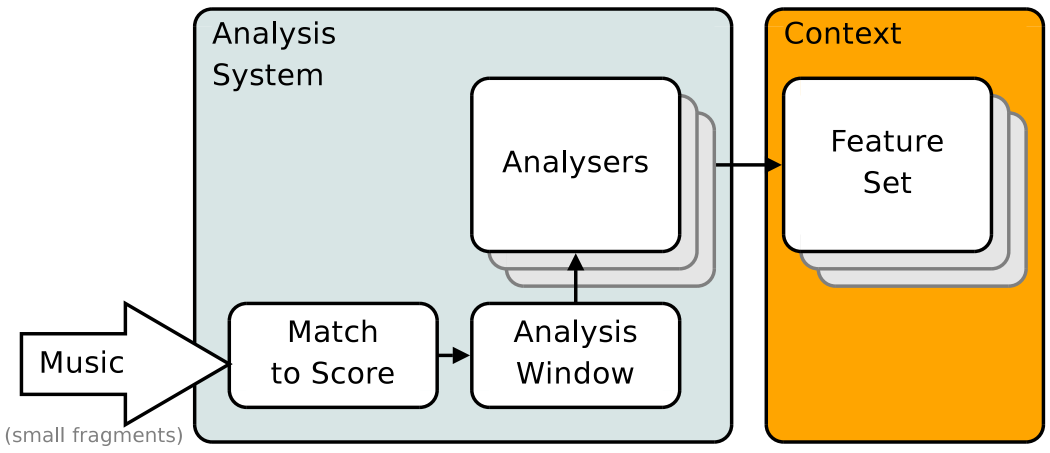


Figure 8.2: Analysis system overview

- any analysis must be amenable to being carried out in real time; it should be computationally inexpensive, and have reasonably constant time and space behaviour under normal conditions.
- as a whole, the analysis system must capture the *interesting* aspects of the music being heard.

Figure 8.2 gives an overview of the Analysis System; in more detail:

- music arrives as small fragments: the size depends on the configuration of the agent infrastructure, but current values tend to be within 0.25 to 2 beats, depending on the trade off between responsiveness and robustness required and the network infrastructure.
- where appropriate, these fragments are matched against the score: the position of every other agent through the score is tracked, and the notes in the fragment are compared to the expected score value, and links are built between the expected and received notes if they match (this matching of played notes to scored notes is necessary for certain analysers).
- these fragments, with or without annotations, are then fed into a windowing system, which ensures that the amount of material the analysers have to work with is independent of the fragment size of the underlying network.
- every available analyser is called, and it runs over the given window of music to produce a value.

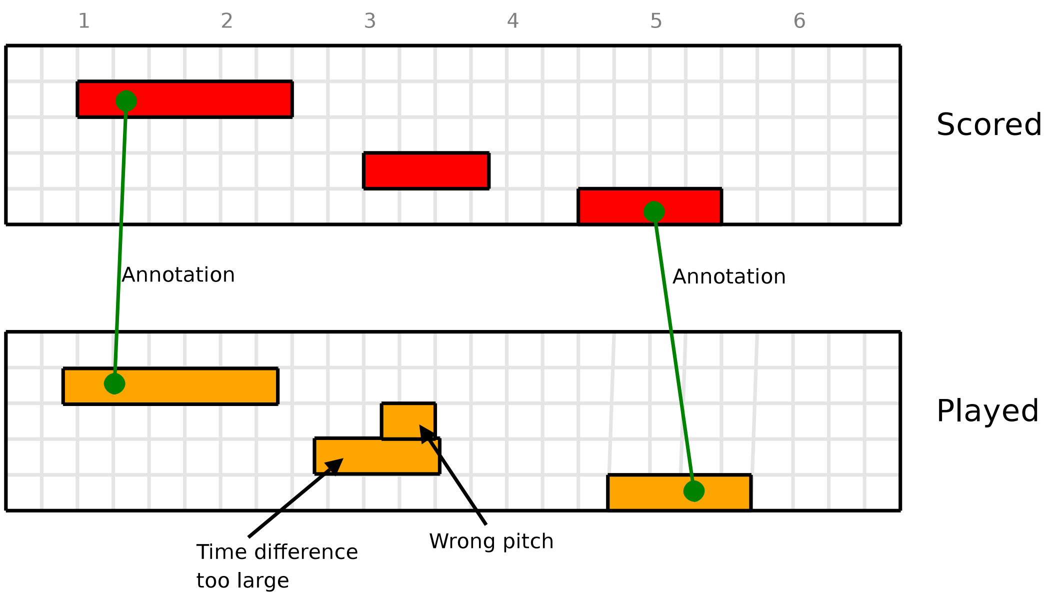


Figure 8.3: Annotating played notes with links to scored notes

- the value for each analyser is stored in a `FeatureSet`, which is part of the `Context` which the agent maintains.

8.2.1 Using the Score

As mentioned above, there is a process which attempts to create `AnnotatedScores`, where each of the notes that has been played is linked to a score note (see Figure 8.3). For every agent, a score following module keeps track of where that agent is in the piece, and can provide the next fragment of notes that agent is expected to play. At present, in order for this to work, the agents must be constrained to playing a fixed score, but this could be improved later. Once the played notes and the scored notes are available, a very simple algorithm is run which matches a played note to a scored one if they are the same pitch, and their onsets are within a certain threshold of each other.

This is only one way in which the score and the context are used; another important aspect is that of quantisation. Some of the features, especially pattern based features rely on a quantisation value; this is the finest level of granularity which scored note positions are expected to take. The value for this quantisation level is recorded in the score, and each agent configures itself accordingly.

It is desirable that these properties — quantisation levels, ideal onset and durations of notes, which are currently read from the score — could eventually be computed by the agents themselves. For instance, human listeners would differentiate between a note which is scored as a crotchet and played staccato and a note which is scored as a quaver. Similarly, agents could be expected to calculate the kinds of metrical grid which best explain notes they hear. Since these abilities are complex, and not necessary for the task at hand, they have not been implemented.

8.2.2 Features

In general the range of possible features which can be analysed is immense; a small subset of features is implemented here, as it is felt that these features can capture a large amount of the information which is useful in the setups used this project. Three aspects of the music under examination are dealt with:

- dynamics are extracted from the `Note` objects
- note onsets are computed relative to "ideal" positions; this is either the closest metrical grid point (the spacing of which can be set via a quantization parameter), or the position of the note this one is linked to, if annotations are available.
- ratios between note lengths and "ideal" lengths are calculated. Here again, either the annotated note length is used, or the generally inaccurate assumption is made that the note was scored as being one metrical unit long.

For each of these aspects, three features are calculated:

- the average level of the values is calculated, by adding the values for the entire window and dividing by the number of notes.
- a simple linear regression is carried out on the values to determine the rate of change of the values - modelling *crescendi*, *ritardandi*, and so on.
- pattern values are calculated from the deviations in value which are not explained by the linear regression model.

Pattern values are designed to model recurring features, in order to pick up systematic variations in timing, dynamics and so on. A pattern analyser:

- is set to be a certain number of beats long, and divide each beat into a certain number of buckets.

- takes a set of values in, and computes their differences from the "predicted" value; in this case, this is provided by the output of a linear regression analyser.
- assigns each value to a particular bucket. Timings are taken modulo the length of the pattern, so that every value falls into a valid bucket.
- computes the average value for each bucket.

Each of these `Analysers` is run, for the output of each agent, producing a `Attribute` of the appropriate type: averages produce `NumericFeatures`, regressions produce `DualValuedFeatures` and pattern analysers produce `PatternFeature`[1]. These are then aggregated into `GroupFeatures`, which also provide an average value for the group, and stored in the `FeatureSet` which is part of the agent's context.

8.2.3 Creating Analysers and Dependencies

It is up to each agent to determine what features to analyse - in general, the available analysers will not be known until runtime. The `AnalysisSystem` hence works as an analyser factory, creating analysers and managing their dependencies. If an agent decides to use a particular analyser, it passes the analyser's name to the `AnalysisSystem`. The system checks to see if that `Analyser` exists, and if not, it adds any dependencies for that analyser, creates it, and adds it to the list of analysers to be called each time a new fragment of music is received.

Dependencies are used to allow the analysers to use each others output. For example, in the previous section, it can be seen that pattern analysers rely on the output of a regression analyser (to obtain predicted values for each note). Each pattern analyser can list this dependency, and the `AnalysisSystem` will guarantee that the relevant regression analyser exists, and that it is run before the pattern analyser.

8.3 Generation

The aim of the generation system is twofold: firstly, it is generally desirable to output music with the expressive characteristics that make it feel "human"; secondly, and more importantly, the output must be capable of conveying the agent's intentions by embodying certain characteristics.

[1]The names of the Java classes do not exactly reflect this for historical reasons, but this is not important here

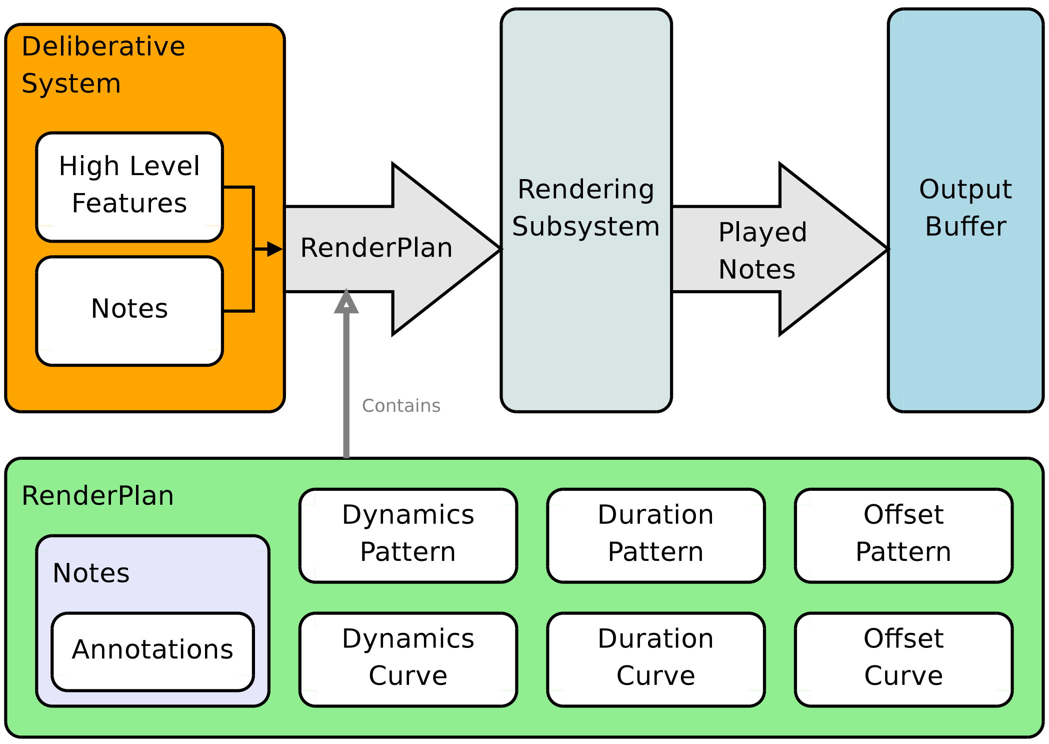


Figure 8.4: Overview of rendering pipeline for producing music

The conception for the generative system is that a set of high level decisions have been made about what to play in general terms, but there is still some work which needs to be done to turn them into music. For example, the agent might have decided to play a walking bass in Eb, using 8th notes, with strong accents on the 2 and 3, and gradually crescendoing. There is now some work which needs to be done; the exact set of notes must be worked out, then each note must be given a precise timing and dynamic value combining all the necessary features. In some senses, the generative system can be seen as similar to the unconscious actions which happen when a player's hands fit themselves around the instrument (see Sudnow [1993] for ideas in this area); although the bassist knows they are about to play a walking bass, they might not know exactly which notes they will play until their hands take over and they start playing music.

Currently, the system is not designed to generate its own notes: the notes to be played must be provided by the score. Given this, the generation system can be thought of as *rendering* a set of abstract properties and notes into a concrete set of notes-as-played, to be transmitted to the rest of the system. The set of high level properties, combined with some notes to play is termed a `RenderPlan`.

The “rendering pipeline” for this stripped down situation is hence (see Figure 8.4):

- high level features from the deliberative system are placed together with notes from the score in a `RenderPlan`.
- the rendering subsystem applies the features to the notes, and places the result in the agent’s output buffer.

8.3.1 RenderPlans and their realisation

As with the Analysis system (Section 8.2), there are three aspects of notes which are affected: onsets, dynamics and durations; there are three ways in which these are affected, but these differ slightly from the previous set. For each of the three aspects, the render plan carries a curve, which indicates a baseline value, and a pattern, which indicates repetitive deviations from baseline. It is also possible for notes to have properties added to them, to signal that they should be accented in some way. In detail, for each of the three aspects:

- the value of the curve at the time of the note’s onset (as scored, not as played) is calculated and applied. Dynamics are given as absolute values between 0 and 1, onsets are given as beats, and duration is a proportion of the scored duration.
- the value of the pattern is calculated for that note; the note’s time relative to the previous “pattern anchor” is calculated, and used to extract a value from the pattern supplied in the rendering plan. Pattern anchors that are used as the start of a repeating pattern may not be the start of the fragment which is currently being generated. At present, patterns are anchored at the start of each Section in the score, but this could be changed by any agent that needed to do something more complex. This pattern value is then added to the value for the note[2].
- finally, any annotations which the notes carry are applied. Currently understood annotations include gracenotes, slurs, legato, marcato; the effects of these have not been rigorously calculated, as they are not used in the current system. The mechanism, however, is in place for later work.

At the end of this, a complete set of notes with onsets, dynamics and durations is produced, and passed on to the rest of the agent system.

[2]The actual code implementation does not follow this exactly: duration is calculated as $d_{played} = d_{scored} * (d_{curve} + d_{pattern})$.

8.4 The Mirroring Reasoner

The mirroring reasoner was created to use in the experiment described in Chapter 11. It is designed to be a "featural mirror" to a human performer, extracting features from the human playing and applying them to the generated playing: as the human starts playing *staccato*, so does the system; as the human plays louder, so does the system and so on. As previously explained, the task of a deliberative system is to produce a Render Plan, which the generation system can turn into notes-in-time. The basic operation of the mirroring reasoner is simple: for every feature extracted from the playing of the human partner, a similar feature is put into the render plan which is later output.

In more detail, some other issues come into play. There are three types of features currently used in the system: averages, curves and patterns, and each of these types is calculated for dynamics, timing and note lengths. When creating a Render Plan, the mirroring reasoner works as follows:

- pattern values are copied across verbatim; whatever pattern of accents or articulation the human used will be echoed by the system as quickly as possible.

- curve values are ignored.

- average values are used to construct a curve in the Render Plan, which goes from the previous average value to the new average value, over the course of one fragment. This means that, for example, volume is smoothly modulated, but matches the player's value as quickly as possible.

8.4.1 Value Reasoner

An alternative version of the mirroring reasoner was used in the experiment; in this case, once features have been extracted, they are converted into the symbolic features used in Chapter 9, and then back to numeric values, before the render plan is created. This is to match the use of symbolic values in the other reasoner used in the experiment.

8.5 Discussion

As it stands, the system operates in a musical microworld: there are several areas where major simplifications have been made. At this stage, these aspects are not necessary, and would add considerable complexity and overhead. However, in the context of a more general agent architecture, it is necessary to note the limitations and discuss whether more functionality could be added in the future. The current system:

- requires a detailed score; there are many times where this would not be possible, even when some score is present. For instance, when playing with a “Fake Book” which gives chord sequences and melodies but not actual notes. On the analysis side, it is clearly possible to choose features which do not need to be linked to a score. Note durations can be calculated absolutely, and inference could be performed about the “ideal” notes that the played notes are derived from, hypotheses could be constructed about metrical grids to fit the played notes to, and so on. For generation, it is certainly conceivable to have a system which comes up with notes based on high level features; the PACTS discussed in Pachet et al. [1996] are a perfect example of this. Another example of a system which works in a similar way to the module is Rubette, described in Mazzola and Milmeister [2006].

- forces adherence to an isochronous pulse; as discussed previously, this could be handled by having another system perform beat tracking, and fitting played notes onto a metrical grid. This might entail some level of communication between the layers, to allow analysis to take account of changing hypotheses about beat placement.

- analyses features which are simple functions of the musical surface. For a system to be truly musical, it would need to be able to analyse more complex properties; for example, the end of Chapter 6 called for analysis of the “musical direction”, and of actions which would complete at certain points in time. Again there is no reason that this system could not deal with these features, but currently nothing is in place to support this.

8.6 Conclusion

This chapter presented an analysis and generation architecture which is compatible both with the theory described in Chapter 6, and with the Musical Middleware specification from Chapter 4. A simple method for constructing musical output in response to a human performer was described, which is later used in the experiment in Chapter 11. Finally, the system currently operates with a restricted feature set, as this “microworld” allows the effects of different components to be clearly seen; however, it is believed that later expansion to a richer set of capabilities would be possible.

Chapter 9

Musical deliberation

9.1 Introduction

Previous chapters have detailed how an agent system can be implemented which can generate music in response to the output of others; however, they have not touched on the way these responses are generated. This chapter describes the creation of a simple deliberative system based on the principles of Musical Act Theory described in Chapter 6.

In a general sense, the task of this deliberative system is to analyse the playing of the agent's peers and:

- look for playing which can be considered a communicative action.
- generate appropriate communicative actions in response.
- use these actions to influence the generation of music.

This is shown with reference to the overview of a musical agent in Figure 9.1.

9.1.1 Design Criteria

The work in this chapter relates both to a theoretical framework for constructing deliberative systems, and to the implementation of a module which is used in the current system. There are a set of criteria which have influenced the design with respect to its uses in the current system, which are:

- the system must work in realtime, so it should be both computationally cheap and consistent in execution time.

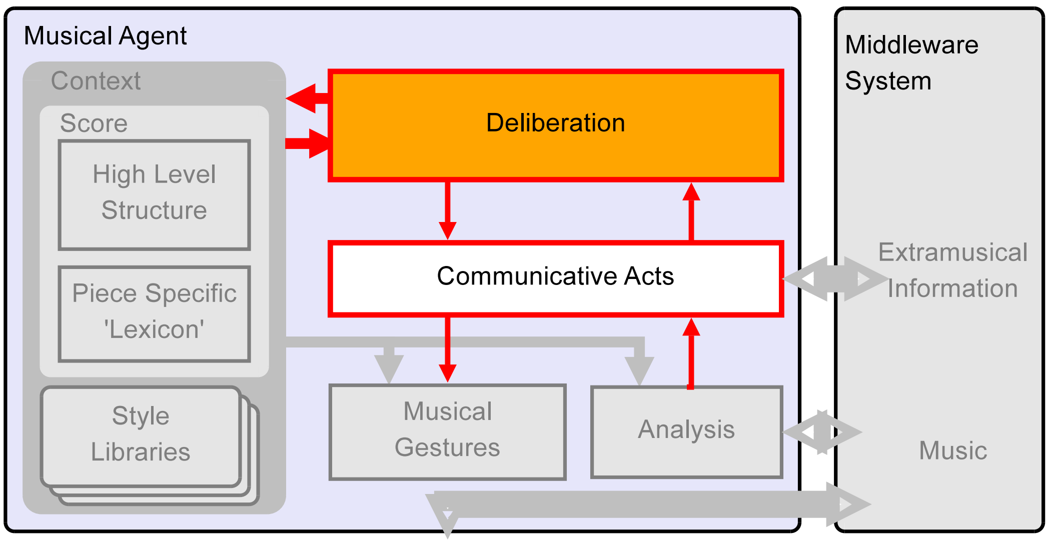


Figure 9.1: Overview of the components covered in the Deliberation chapter

- the operation of the system should be understandable, to aid debugging. This applies both to whether the system is functioning as designed, and whether the design is appropriate for the task.

- the system must be able to work with a relatively limited amount of data, as this is not freely available.

9.2 Deliberation formalisation

The first task with the design of the module is a description of the states involved in its operation. The deliberation system needs to provide actions for the system to enact. In general, this means outputting a series of musical actions, which lower levels can then render into features, and finally notes.

In this formulation, Musical Actions are used, as distinct from Musical Acts: the intentional aspect of musical acts is not used. A musical action details the relationship of a new piece of playing to the current musical context, so the job of the deliberation system is to produce a set of relations which should hold for the next chunk of playing, based on the relationships between previous playing, and on previous actions.

A state description for events in a musical agent system has already been carried out in 6.4.3; however, the system here will use a slightly different formulation, for two reasons: Finally, this formulation is for the simple case of two agents playing together;

the possibilities for generalisation to many agents will be discussed at the end of the chapter.

9.2.1 State formulation

Based on a the notion of a state indexed model of music which has been discussed in Section 6.4.3, the output of a pair of musicians can be thought of as a series of states, where states change every time one agent's value for a facet changes.

Each facet of music will be considered independently, so that the overall state will be made up of a set of sub states which may change at different times; this simplification is made so that each facet can be reacted to individually, without being dependant on events in other facets.

A further simplification is made — that the actions which cause state change alternate between the agents. This is not necessarily true, so the analysis will insert "dummy actions", so that if agent A performs two actions, a dummy action by agent B will be inserted between them. This is done to simplify deliberation, as actions can always be treated as an alternating sequence.

The only relations considered here are between a new state and values in the previous state; this may appear somewhat similar to the assumption made for a Markov process, but this is not the assumption being made here. Instead, longer term relationships will be dealt with by another part of the model. Figure 9.2 shows the values and relations between them through a progression of states, considering only R_{self} and R_{other}.[1] The repeated values are not shown, and v_s^k represents the value for agent $self$ in state S_k.[2]

When examined in terms of states and actions, the picture becomes a linear sequence of states, with the edges between states labelled with the action which causes the state transition. Figure 9.3 shows these states, which have been arranged to indicate which agent caused the new state, with action leading to state S_k labelled as A_k.

Each signature here is then a tuple (R_{self}, R_{other}), with both values coming from the

[1]In Section 6.4.5, three relationships were considered for each new state ($R_{self}, R_{other}, Rprev$). The reasons for only considering two relations here are:

- the system of musical acts was still under development at the time this module was created, and so later changes were not taken into account.
- the formulation here relates more cleanly to the n-gram sequence model which is used.
- the reduction in alphabet size helps to deal with the small amounts of data available.

However, Section 9.4.2 discusses why this formulation was altered to that given in Section 6.4.5.

[2]This is not entirely consistent notation, as subscripts were previously used to indicate state indexed- rather than fragment indexed- time.

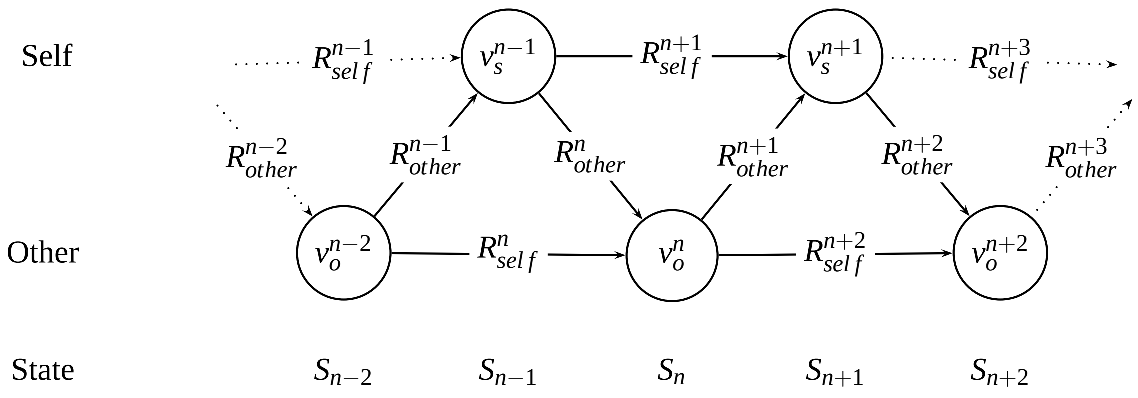


Figure 9.2: State Transitions in Dialogue Formulation with Relations, with only novel values shown

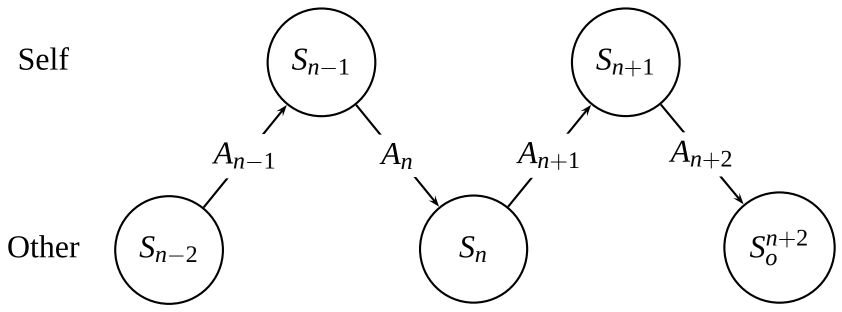


Figure 9.3: Musical Action Signature State Diagram

set of lattice relations from Section 6.3.2, i.e. `SAME,SUBSUMED,SUBSUMES,ALTER,DISJOINT`. This gives an alphabet of 25 possible signatures.

9.2.2 Role of the deliberator

The system model being used is hence a sequence of states $S_1 \ldots S_n$, caused by a sequence of actions $A_1 \ldots A_n$; when an agent decides to take action, and create a new state, the task is to choose a new action, and then execute it to create the new state. The role of the deliberative system is to choose A_{n+1} given $A_1 \ldots A_n, S_1 \ldots S_n$. In the general, this can be done by any appropriate method; for example some kind of interaction protocol could be used to generate new actions, internal goals could be used or a human user could be continually specifying actions. The rest of this chapter details a simplified case, in which A_{n+1} is chosen based on sequence completion, supported by a database of previous human-human interactions.

9.3 Deliberator System Design

The previous section detailed a model for the task that the deliberative system needs to perform. This section integrates this model with the rest of the agent system, and describes an implementation.

In order to integrate with the rest of the agent system, an architecture is used as shown in Figure 9.4. The main components of this are:

Act Extraction takes the high level descriptions of music produced by the Analysis system, and extracts musical action signatures (MAS) from them.

Deliberation takes the stream of MASs and produces new MASs in response.

Action Realisation takes the generated MASs and produces high level descriptions which can be put into a `RenderPlan` and passed to the Generation system.

9.3.1 Symbolic Values

The analysis system produces values for features; currently, these values are represented using floating point numbers: they are either a single numeric value or a structured set of numeric values. This allows for the representation of a wide range of data, and also makes few assumptions about the ranges which the values take. When

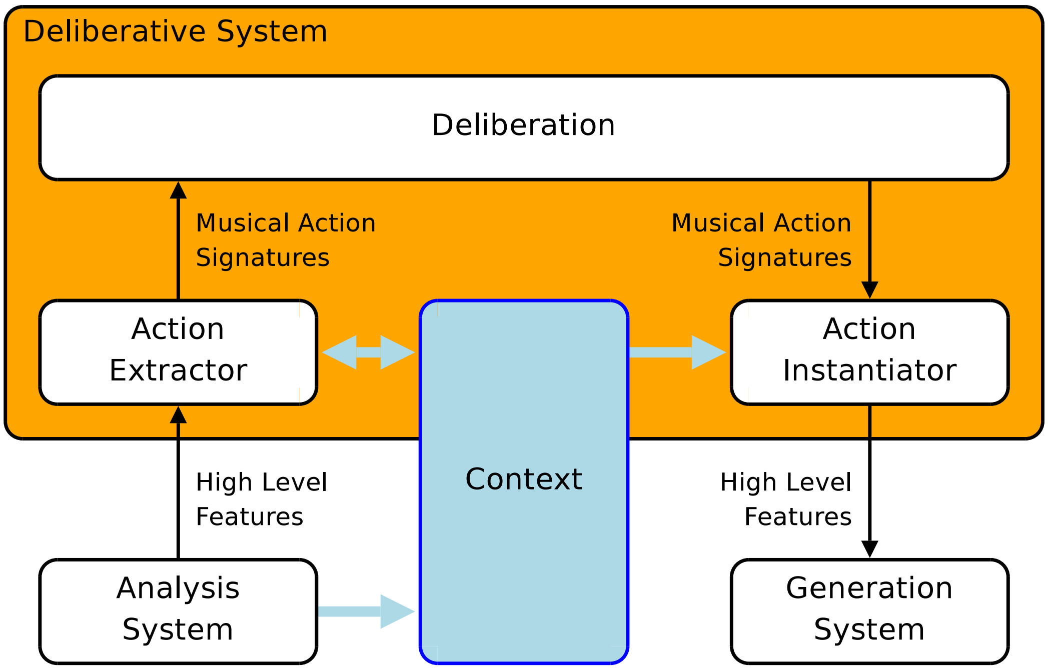


Figure 9.4: Overview of the deliberative system architecture

it comes to deliberating about values and situating them on a lattice, it is easier if the domain of the values is finite, so a layer of symbolic values is introduced. For every numerically based feature whose value is to be deliberated about, it is necessary to define:

- a set of symbolic values on a lattice,
- a process to map numeric values onto symbolic lattice values,
- a process to convert the symbolic values back to numeric values.

9.3.1.1 Spectrum Values

First, there is the issue of how to convert scalar numeric values to ones which may be placed on a lattice. The approach taken here is to assume that any particular numeric value has a “default” or “natural” position. This is equivalent to the level which a player would assume for that feature if no markings were given in the score; for example, dynamics would be around *mf*, note lengths would be about three-quarters of the notated values, and notes would be placed exactly on the beat. This default position is then taken to be the top of the lattice ($\odot$). From this root, two branches extend: one where the value increases, and one where it decreases. At the extremes, when the value cannot be increased or decreased further, it is impossible for music to realise these values, so the bottom of the lattice is reached ($\ominus$). Figure 9.5 gives an example of a generic numeric lattice, and a configuration for representing dynamics. These values are termed spectrum values.

9.3.1.2 Pattern Values

As well as simple numeric values, the system also contains composite values: lists of numbers which represent repeating patterns. The symbolic equivalents of pattern values start from the numeric pattern values, and replace each numeric value in the pattern with a symbolic spectrum value. The lattices used for the spectrum values need not be the same as the values which are used to represent the underlying property that the pattern is found in; that is, the lattice used for values in patterns describing variations in dynamics would not be the same as the lattice used to represent average dynamic values — the example in Section 9.3.1.4 may help to clear this up.

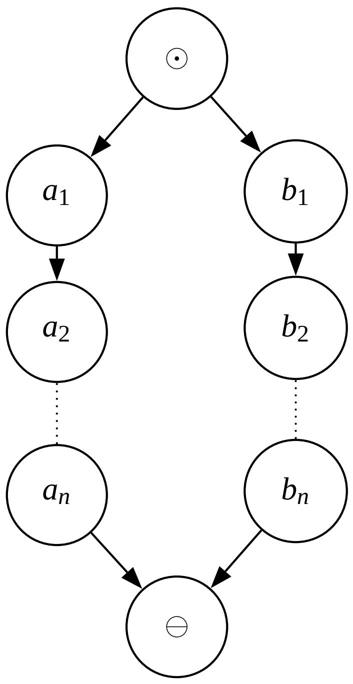
9.3.1.3 Implementing system values

One of the issues with symbolic values is that a numerical definition of the symbols must be given to allow this conversion to take place. In this part of the system, there are three aspects of music under inspection: dynamics, onsets and durations of notes. For each of these, the average value is of interest, as is a "pattern" value (see Chapter 8). These values all have different ranges — for example, dynamics are between 0 and 1, while timing can be positive or negative, with no definite upper or lower bounds.

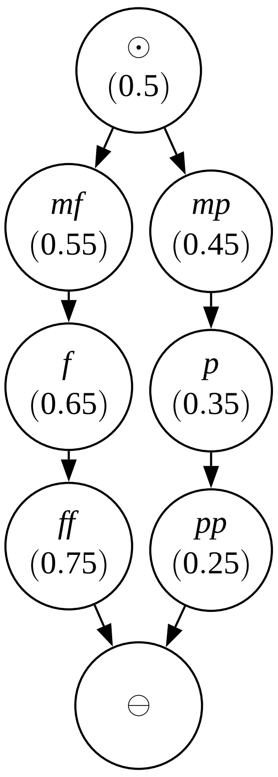
There is a choice about the number of symbols used for a particular value: are they tied to common concepts (e.g. musical terms)? what numbers are used to define them? These choices are considered to be entirely implementation specific - it is quite possible for a group of agents to have different values for their symbols so long as they do not attempt to communicate explicitly using them. Agents could also adjust the number and values of symbols they use as a piece progresses, to find the set which most usefully describes the playing of others; they might even maintain different lattices for analysing the output of different musicians.

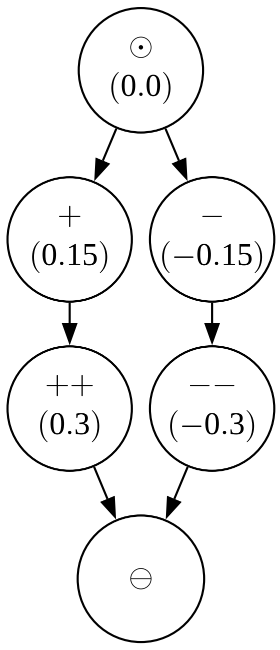
In this particular implementation, the lattices are constructed on an ad-hoc basis, as follows:

Dynamics are modelled using standard musical terms, as shown in Figure 9.5b, while dynamic patterns are modelled using the lattice shown in Figure 9.5c.



(a) Generic numeric value lattice





(b) Numeric lattice for average dynamics values

(c) Numeric lattice for dynamics pattern values

Figure 9.5: Example numeric lattices

Note lengths are equated to legato, long, short and staccato. These values are relative to their scored length, so this would come under the heading of articulation.

Note timings can be early, earlier, very early, late, later or very late, again relative to their scored positions.

These lattices are maintained by a class which implements `LatticeManager`, allowing any agent to substitute a specific set of lattices if desired — this is important, as it is not intended to force a particular set of lattices to be used — the values used here are simply one possible setup.

9.3.1.4 Example

As an example, consider assigning dynamic symbols to a piece of music. The Analysis module (Section 8.2) has found the average dynamic value to be 0.68, and the average patterned deviation from this value to be $(0.32, 0.05, -0.15, -0.12)$.

Starting with the average, it will be assigned the most "extreme" value — furthest from $\odot$ — which is less extreme than the numeric value value. For example, $\odot$ is 0.5, *mf* is 0.65 and *f* is 0.75, so the average dynamic of 0.68 will be classed as *mf*. Accents are then modelled using the lattice in Figure 9.5c, so the symbolic version of the pattern would be $(++, \odot, -, -)$.

9.3.1.5 Computing relations between lattice values

It is necessary, given two lattice values, to be able to compute the relationship between them. If the lattices used were fully explicit, this could be performed by walking the graph representing the lattice to calculate ancestry. For spectrum values, this is possible, and works as follows:

- if A is an ancestor of B, A subsumes B, and vice versa.
- if there is a common ancestor between A and B, then A is an alteration of B.
- otherwise, A and B are disjoint.

For pattern values, a more complex approach must be taken, as enumerating the entire graph is impractical. For the values A and B:

- the *meet* of the two values is calculated, by calculating the pairwise meet of each value in the two patterns.

- if the meet is equal to value A, then B subsumes A, and vice versa.
- if every value in the meet is $\ominus$, then A and B are disjoint.
- otherwise, A is an alteration of B.

9.3.2 Act Extraction

Act extraction is driven by the analysis system; each feature which is analysed may have an `ActExtractor` attached to it. The design of the act extractor is simple:

- when started, the extractor asks the `LatticeManager` for an appropriate lattice for the feature it is working on.
- every time new music is received, the analysis system generates a new value for each of the musicians involved. The act extractor uses its lattice to symbolise each of these values.
- each value is compared with the current value for that agent; if it is different, that is counted as an action.
- when an action is found, the relation is computed between the new value and:
  - the previous value for that agent.
  - the current value for the agent doing the analysing.

  Both of these are maintained in the `Context`. A `MusicalAction` object is created, containing both these relations, the new value which has been found, and the time at which the event occurred.
- for each timeslice, all of the actions which have been found are passed up to the Deliberation module, and also stored in a history.

9.3.3 Deliberation

As stated before, the deliberation module is designed to be as simple as possible, both conceptually and architecturally, and have well behaved operation. The operation has been formulated as choosing S^n given $S^1 \ldots S^{n-1}$. One possible way to carry this out is to have a database of sequences (of musical action signatures) and their frequencies, and find the most frequent completion for the current sequence. This is attractive because it allows a very simple data-driven approach, without requiring any further explicit knowledge about interactions. In implementation, this is carried out as follows:

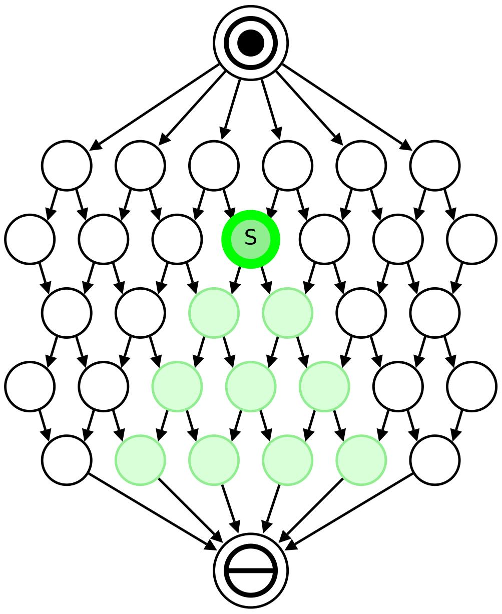
- the `SequenceManager` provides a set of `SequenceTrees`. Each sequence tree accepts a sequence and returns a set of possible completions along with their frequencies. Each tree accepts sequences of a particular length, and a `SeqeunceManager` of order N will return a set of trees of order $1 \ldots N$.
- When a new action is passed to the deliberator, it:
  - retrieves the previous sequence of actions carried out by that agent for that feature.
  - extracts the last N actions from the sequence, and attempts to find this in the tree of of order N.
  - if the sequence has possible completions, one of these is chosen probabilistically according to their frequency.
  - if no completions are found, a sequence of length $N-1$ is tried, until a zero length sequence is reached, at which point a default response will be chosen.
- the chosen action signature is then passed to the Realisation system.

It can be seen from this that:

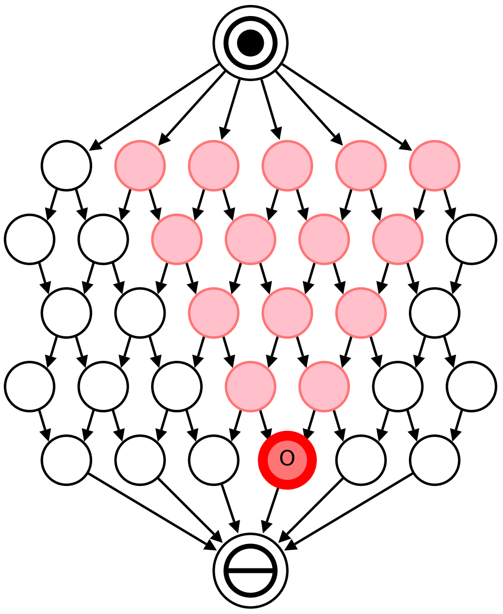
- each feature is treated completely separately — that is, an action extracted for one feature will not cause an action to be emitted for another feature, or influence the choice of action if one is already being emitted.
- a database of sequences must be provided. This is discussed in Section 9.3.5.
- operation at run time is well behaved; at most, N sequence lookups are performed before a chosen action is returned.

9.3.4 Instantiation

Once a particular action has been chosen, it must be instantiated. The action signature defines relationships between the new value for this agent, and its previous value, and the new value and the value from the other agent which precipitated this action. A value must be found which satisfies both these conditions.



(a) Possibilities for `SUBSUMED` and *s*



(b) Possibilities for `SUBSUMES` and *o*

In general terms, the combination of a value and a relationship defines a portion of the lattice; on a finite lattice this is the same as the set of values which hold that relation to the given value. If two relation-value pairs are needed, then the intersection of the sets produced gives the set of values which match both constraints. It is then up to the particular implementation of deliberative system to choose one of these values. To clarify, imagine that r_s is to be `SUBSUMED`, and this agent's current value is s; the possible choices are shown in Figure 9.6(a). If r_o is to be `SUBSUMES`, and the value for the other agent is o, the possibilities are shown in Figure 9.6(b). Putting together these two constraints gives us the possibilities shown in Figure 9.6.

9.3.4.1 Implementing value choices

At this point, it becomes necessary to be able to both derive the relations between lattice values, and construct a set of values from a value and a relation. This would be relatively trivial given a finite lattice where the complete graph is known, but in this system this is not always the case.

With the numeric "spectrum" values as demonstrated in Figure 9.5 (where the lattice consists of two continuous chains from top to bottom with no interconnections), this approach is possible. Given a value and a relation, the appropriate set is generated by:

SUBSUMES: the ancestors of the current node.

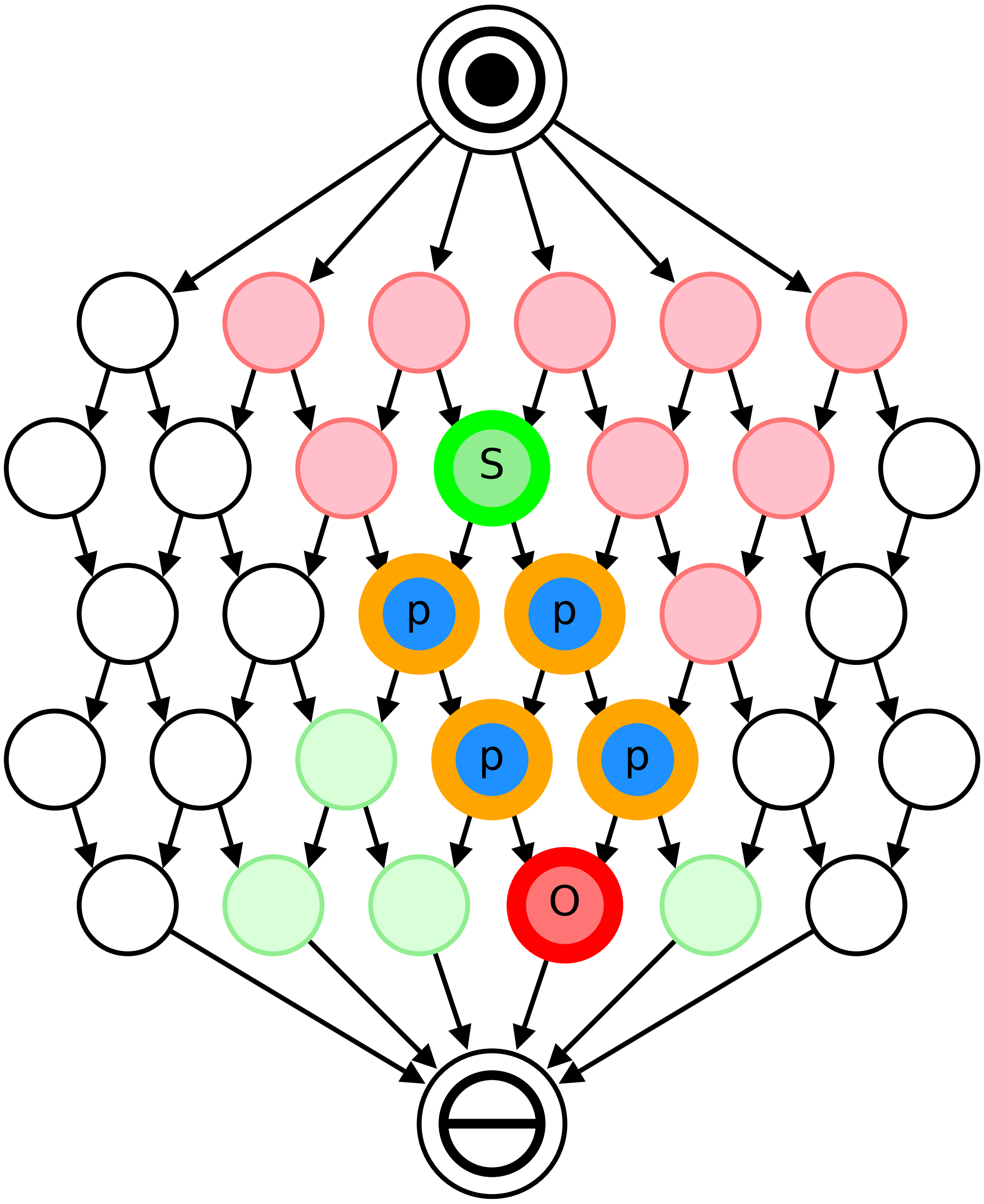


Figure 9.6: Combination of the two constraints on next value (nodes marked p are possible)

SUBSUMED: the descendants of the current node.

ALTER: the empty set.

DISJOINT: the other branch of the lattice.

For the pattern lattices the set of possible values is very large; for example, with a 10 step lattice representing dynamic accents (see Figure 9.5c), each step has 4 possible values, plus $\odot$ and $\ominus$, giving a possible space of 6^{10} values for the whole pattern. It would not be possible to perform set operations on even a portion of these sixty million values, so an alternative approach is taken:

Generating a random new pattern with r_a to pattern value a

- for each step of the pattern, the set of values which would be appropriate for that step is calculated using the procedure given above for spectrum lattices. Then for each step of the pattern, appropriate values are those which have r_a to the current value for that step.
- a new pattern is generated by sampling each of the possibility lists in turn.

To construct a set of possibilities for two value-relation pairs

- a sample is generated having relation r_a to value a,
- this sample is tested for having relation r_b to value b,
- if this succeeds, it is added to the set of possible samples.
- repeat, alternating which value/relation pair is sampled from and which is tested against, until the desired number of samples is reached.

For both pattern values and spectrum values, once a set of possibilities is generated, a random value is chosen from the set. If the set is empty, then the value of the agent which triggered this action is used.

9.3.4.2 Numerisation

The symbolic values produced by the deliberative module must now be converted into the numeric form used throughout the system. There are many possibilities here, but the simplest strategy is currently used: for each value, the number associated with each symbol is used to create a numeric equivalent. These numeric values can then be put together into a `RenderPlan`, and send to the generation system for rendering into music.

9.3.5 Filling the RenderPlan

The deliberation module must now construct a RenderPlan, in a similar manner to Section 8.3.1. That is, pattern values are inserted directly into the plan, and average values are used to create a curve from the previous average value output to the new average value, which lasts the length of the RenderPlan.

9.4 Training the reasoner

In order that the deliberative system behave as much like a human musician as possible, it is seeded with a set of musical action signatures which are extracted from human playing. This system used a small corpus of data, which was collected as described in the following paragraphs.

9.4.1 Data Capture Setup

The sequence data was intended for use in the experimental setup described in Chapter 11, so it was collected using a similar setup; this should help ensure that the same type of human behaviour is captured in the sequence tree as would be expected in the experiment. Briefly:

- two pianists, who had not previously worked together, were used; they were seated on either side of a screen, wearing headphones and playing electronic pianos.
- each pianist could hear their own playing, the other pianist's playing, and a metronome.
- deviating from the experimental setup slightly, the pianists were asked to repeat certain sections from Canto Ostinato continually until they both stopped. This meant that they did not have to be counting bars or trying to read the next notes to play, so all of their attention could be focused on the interaction with the other player.

- these duets were saved as MIDI files. The obviously erroneous parts of the files were removed, and the results fed into the analysis system for act extraction.

9.4.2 Analysing and Visualising the data

The analysis system used here is the same one used by agents in the agent system, with a few small additions:

- since the analysis is between two agents, rather than between "self" and several other agents, dummy agents are created, and the interaction between them is analysed.
- only the analysers which are used by the deliberative system are loaded.

The analyser carries out the steps involved in creating a sequence of actions for the deliberative system, namely:

- creating symbolic versions of the numeric features extracted.
- looking for changes in these symbolic values.

The analyser maintains an idiosyncratic context in order to calculate relations between the actions of these dummy agents, and uses this to produce a stream of fully defined musical action signatures from the MIDI file. Once the whole file has been processed, the analyser feeds the two signature streams into a `SequenceManager` to create a full database of sequences. The two sequences are treated as being part of the same data set, and simply added sequentially to give one overall database. This database is then written to a file (as a serialised Java object) for the agent system to use in its deliberation.

In conjunction with this, a visualisation tool is provided, which displays the output of the two participants, along with the acts which have been extracted. A screenshot is given in Figure 9.7.

9.5 Feedback from experimental trials

Chapter 11 details an experiment which compared the perceived interactivity of the MAMA system using either this deliberative module or a simple module which mirrored the playing of a human participant. Traces of these experiments were saved, so

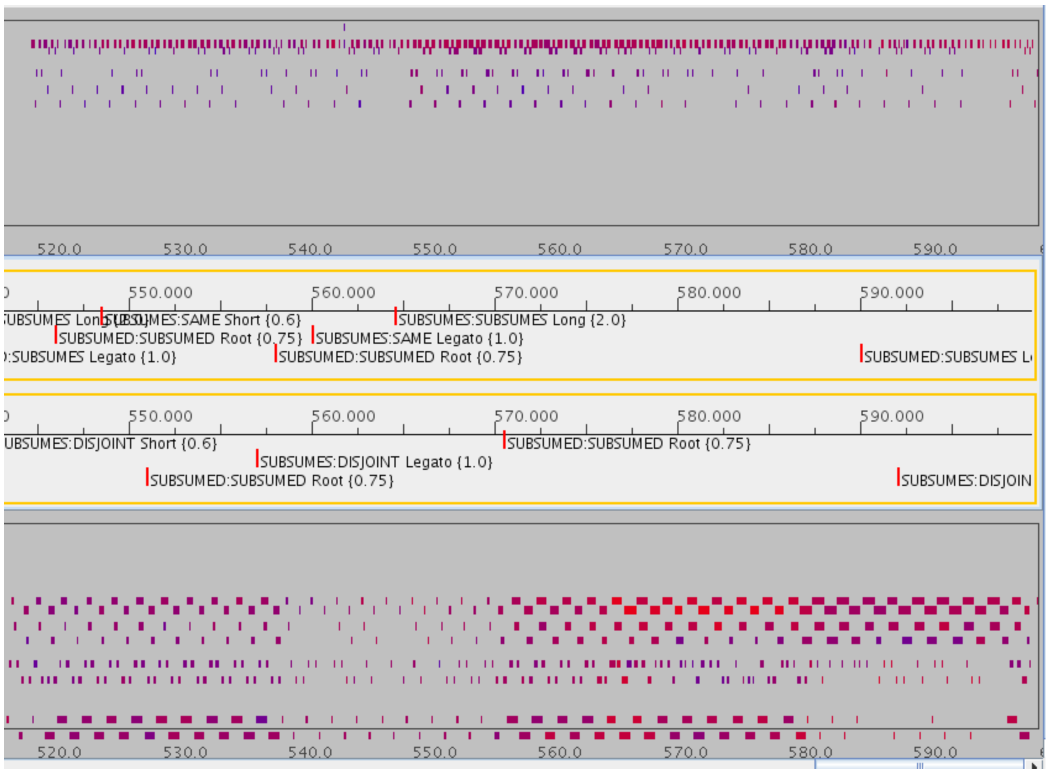

520.0
530.0
540.0
550.0
560.0
570.0
580.0
590.0
550.000
560.000
570.000
580.000
590.000
UBSUMES Long
SUBSUMES:SAME Short {0.6}
SUBSUMES:SUBSUMES Long {2.0}
SUBSUMED:SUBSUMED Root {0.75}
SUBSUMES:SAME Legato {1.0}
:SUBSUMES Legato {1.0}
SUBSUMED:SUBSUMED Root {0.75}
SUBSUMED:SUBSUMES L
UBSUMES:DISJOINT Short {0.6}
SUBSUMED:SUBSUMED Root {0.75}
SUBSUMES:DISJOINT Legato {1.0}
SUBSUMED:SUBSUMED Root {0.75}
SUBSUMES:DISJOIN

Figure 9.7: Visualising the actions extracted from playing

that the actions of the reasoner could be discussed in a more general context. However, upon examining these traces, a worrying result emerged: of all the times where an action could be taken, the value generated for that action was `null` approximately 85% of the time. This meant that in the experiment, the system was behaving mostly in the same manner as the simple mirroring module.

The first cause for this was found to be in the manner of selecting values given two value-relation pairs. When looking at one of the pattern features, the space of values contains of the order of 10^7 possibilities. Each value-relation pair defines a subset of this space. The original algorithm for generating a value from two value relation pairs worked as follows:

- generate the two subsets of potential values, one from each value-relation pair.
- calculate the intersection of these sets.
- choose a value from this intersection.

This quickly proved to be impossible: the time alloted to this is fractions of a second, so generating even half of 10^7 values is not an option. A modification was made, on the assumption that most of the time these subsets were likely to be widely spread, and there would be a good chance that a value from one set would be in the other:

- randomly sample each space 100 times.
- calculate the intersection of these two sample sets.
- choose a value from this intersection.

This is the algorithm which was used in the experiment. However, it too has a major problem: the chance of any one value being in both of the sample sets is vanishingly small: making the simplification that we are looking for *any* of the samples being the same, taking n samples from a space of size s, this can be approximated as the chance of two samples being different multiplied by the number of sample pairs:

$$1 - \left(\frac{s-1}{s}\right)^{C_n^2}$$

With $s = 10^7, n = 100$, the chance of having any two points the same is approximately 1 in 1000. In fact, analysing the results[3] shows that a value was being chosen approximately 15% of the time. This shows that often the space was smaller than this — the spectrum values used for average levels of dynamics, timings and lengths have a space of between 6 and 10 — or that sometimes sufficiently small subsets of the space were being selected that matches could be found. In response to this, the algorithm was changed, to that given in Section 9.3.4.1, which is:

- sample from one value-relation pair, and test this sample for membership of the other subset.
- repeat, starting with alternate value-relation pairs until a result is found.

Making this modification brings the proportion of instances where value is found up to 35%, which raises the question: what is happening in the other 65% of cases? In order to explore this, another modification was made to the code: when looking for a new value, the relation between the previous values was printed out. It became clear that there were certain configurations of values and relations for which no satisfactory value could be found; for example if the action is `(ALTER,SAME)`, but the values are

[3]to verify this, the frequency of occurrence of the string "Value is:", and "Value is: null" in the output logs was compared.

currently the same, it will be impossible to find a value which satisfies all of these conditions.

After some reflection, it can be seen that there are many times when there is no intersection at all between the subsets defined by two value-relation pairs. This prompted the discussion in Section 6.5.3 (and Appendix B) about what combinations of relations are possible.

9.6 Discussion

This section has presented a formulation of a model for deliberating about what action to take next in a musical interaction, and the implementation of a particular embodiment of this.

Discussing the model formulation first, the first note is that a future version should be brought into sync with the formulation of musical acts discussed in Chapter 6. The main effect this would have is to change the form of the musical action signatures which are reasoned over. The reason that this has not been done here is that this is the formulation used in the experimental work, and it might become confusing to present an alternative formulation in this context.

There are several qualities which are ignored by the model at present:

- the time between actions is not used in calculation, and hence there is no ability to use the timing of other's actions to gain more insight into their playing, or to plan the timing of one's own actions. This could be attacked by adding some kind of interval representation to the musical act signatures used, although this would have the effect of increasing the space of possible signatures, making the database sparser. In turn, using a larger corpus of data would help, as would allowing some form of fuzzy matching between signatures.

- the relations between actions carried out in different features is not taken into account; it would be reasonable to suspect that agents would vary several parameters of their playing in some kind of relationship — for example if the agent is varying some expressive performance axis, this might affect dynamics, note lengths and other timbral features. There are several possibilities to include this information, but the one most in keeping with the musical acts approach would be to create an analyser which could extract that expressive performance axis from the data. This keeps the deliberation simple, and does not force any particular model of interactions between values — agents could have their own particular analysers for different expressive performance axes.

- only a pair of agents is dealt with. Expansion to multiple agents could be done in several ways:
  - running several separate streams, and choosing S^n separately for each.
  - using the ideas about musical common ground from Sections 6.4.5 and 6.5.6 to construct subsets of the group which can be responded to as one agent.

Similarly, there are several issues to explore in the current implementation of the system:

- the system is sensitive to the choice of values used in the symbolic lattices which are reasoned over. At the moment, these values are constructed *ad-hoc*, which works for this prototype, but could be expanded on in future. One possibility for this is to draw on studies of human playing, and create links between the symbols used to describe music and low level features of the music. This has limitations, however, in that:
  - it only works for features which have symbolic descriptions.
  - the terms used may not be the most useful divisions for agents to use.

  It may therefore be a better strategy to allow the agents to learn their own categorisations, and dynamically adjust their lattices while playing. Although this would be more computational work, it would result in more flexible and adaptive agents, and reduce the amount of human legwork which must be done to introduce new features for analysis. A final possibility would be to allow the system to use continuous lattices, so the stage of symbolisation is not necessary.

- the way in which values are chosen to embody actions is currently not very intelligent; a value is chosen from the range of *valid* possibilities for the dialogue formulation, but with little regard to how musically appropriate it is. Although these two things are not completely disjoint, a more intelligent selection process could probably generate better music.

- the system cannot initiate actions; it may respond to the actions of others, and it may do this in a seemingly creative manner, but it does not currently start off chains of actions and responses on its own. A version which could integrate notions of intention, boredom, personal preferences and so on to take a more active role in pushing the music in certain directions would be a logical next step. However, relative to the concept of musical middleware, the current formulation is quite appropriate — it provides interesting responses when asked to do so.

9.7 Conclusion

This section has presented a formulation for the way in which the playing of agents in a musical interaction is related, and used this to design a deliberative technique. This technique has been implemented as part of the agent system, and trained using recordings of human performances.

Chapter 10

Case Studies - uses of the MAMA system

This chapter describes three case studies which have been carried out using the system. There are several reasons for this:

- by creating a system which realises the Musical Middleware architecture proposed in Chapter 4, the architecture is shown to be a viable template for creating real systems, and can be refined in response to issues encountered.

- to demonstrate that MAMA is a flexible system, and not constructed around a single piece or application.

- to explore different methods of interacting with musical agents, in particular by using the system in the different configuration suggested in Section 4.

- to present work which does not fit into the more rigorous structures used in the rest of the thesis.

This chapter has three sections, which correspond to three case studies. “In C” explores collaboration between agents (with human oversight), “Canto Ostinato” looks at musical communication between agents and humans, and “AgentBox” explores the use of multimodal communication to enhance the experience of human/agent interaction. Each case study is followed by a short analysis of the qualities demonstrated in that particular application.

10.1 In C

This section details the use of the system so far to execute “In C”, a seminal minimalist composition by Terry Riley. It discusses how the score was encoded, and what portions of the system were used to support this. This score was chosen as an example of a highly distributed piece; apart from a shared pulse, there is no defined central authority making structural decisions[1].

This piece was chosen as an example of a score which allows a high degree of autonomy for individual agents; this facilitates the exploration of the Composition and Performance paradigms from Section 4, where a score is created that is then performed by the system, with some non-musical interaction from one who is familiar with the piece.

In order to implement the piece it was necessary to:

- encode the score for the agent system.
- develop the behaviours required to play the score.
- allow the user to control the behaviour of the system.

10.1.1 The piece: In C

Composed in 1964, “In C” was instrumental in the beginning of the Minimalist movement. The piece consists of a list of 53 snippets of music, which are to be played in sequence. Each snippet is repeated an undefined number of times, with each performer deciding when to begin playing the next section. Performers may take breaks, omit sections they cannot play, and have control over dynamics and tempo. Some of the key directions are: (taken from the score, which is reprinted in Appendix C)

- It is important not to hurry from pattern to pattern, but to stay on a pattern long enough to interlock with other patterns being played. As the performance progresses, performers should stay within 2 or 3 patterns of each other. It is important not to race too far ahead or to lag too far behind.

[1]However, I recently had the pleasure of attending a performance of “In C” in which Terry Riley was playing; it became clear by watching the body language of the players that Terry was playing a strong role in guiding the shape of the piece (and also in moving beyond the score given). It should hence be observed that the lack of defined roles in the score does not always mean that none are taken in performance.

- Patterns are to be played consecutively with each performer having the freedom to determine how many times he or she will repeat each pattern before moving on to the next.
- Each pattern can be played in unison or canonically in any alignment with itself or with its neighboring patterns.
- The group should aim to merge into a unison at least once or twice during the performance.
- IN C is ended in this way: when each performer arrives at figure #53, he or she stays on it until the entire ensemble has arrived there. The group then makes a large crescendo and diminuendo a few times and each player drops out as he or she wishes.

This shows a range of types of behaviour; there are behaviours which are entirely up to the individual - where to place one's pattern with respect to the rest of the group, behaviours for individuals which are dependent on the actions of the group - making sure that one stays within a few sections of the group; and there are behaviours which call for entire group actions - reaching a unison at least once in the piece. A full implementation of the piece would hence touch on many areas of a musical agent system: the balance between autonomy and group coherence, dealing with dynamic structures, analysing the output of other agents and adjusting output to fit the current musical context.

10.1.2 Encoding the Score

A portion of the final score of In C as encoded for the agent system is presented in Appendix C.

Encoding of this piece, just as in the traditional score, takes two parts: specifying the structure and content of the musical material, and then detailing how it is to be used.

The musical structure is encoded with a `Section` for each of the 53 sections given in the score, named s1 - s53. Each of these has a `Channel` called Notes, which contains the notes specified in the score. A blank section s0 is added at the beginning, to allow the agents to start playing at different times. Finally, section 53 is split into two parts; 53a repeats until all the agents arrive there, and 53b is used to add the crescendo/diminuendo and dropping out behaviour. Sections s0-s52 all specify that they derive from a base section, which is outside the main tree. This section is used to specify some attributes and behaviours which are used for all these sections.

10.1.3 Group Following

This direction in the score that "...performers should stay within 2 or 3 patterns of each other. It is important not to race too far ahead or to lag too far behind." is perhaps the most significant direction of all, as the manner in which performers move through the piece is responsible for a lot of its character. In order to do this, the `RepeatUntil` directive is used; this specifies that each agent should repeat the current section until a certain condition holds. In this case, the condition is `followLocus`, which depends on tracking which sections the other agents are playing, and works as follows:

- when an agent moves on to the next section, it sends a message to the other agents to inform them of this. Each agent keeps track of where the other agents are.

- indices are calculated for each section that each other agent is playing - the fringe of the representation tree is iterated over until the relevant section is found, and its index recorded.

- the relative indices are computed, weighted by distance and summed to give a total weighting for moving on to the next section:

$$w_i = \sum_j \frac{w_i - w_j}{1 + d_{i,j}^2}$$

- this weight is turned into a probability using a sigmoid function; the composer/performer can specify

  - the base probability - when the weighting is zero - of moving onto the next section

  - the weighting which is needed for a 0.95 chance of moving on.

By adjusting these variables, the composer can choose how quickly the agents move through the piece, and how tightly grouped they are as they do so. They could potentially be specified for each agent, to allow agents to have a little "personality".

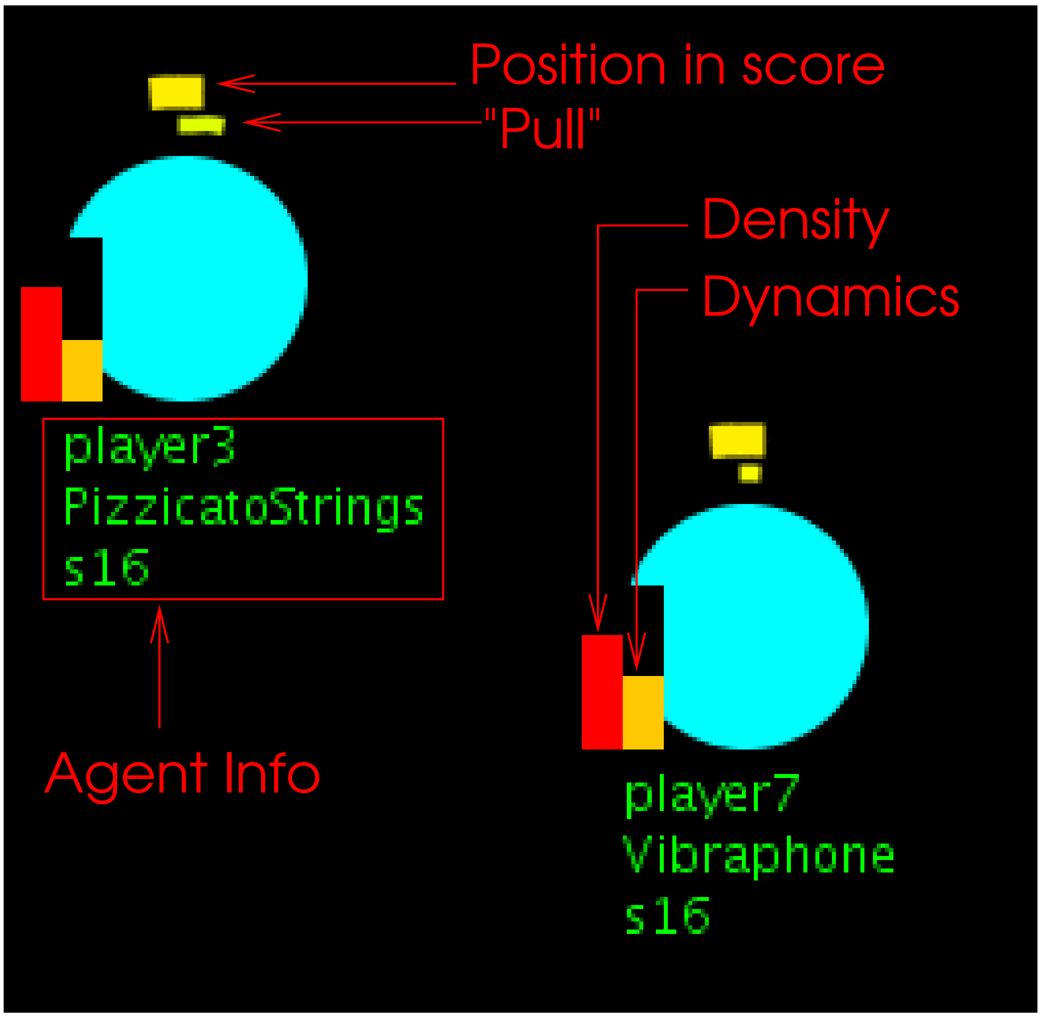


Figure 10.1: Agent Space explanation

10.1.4 Interface: AgentSpace

The realisation of In C uses the AgentSpace interface (described in Section 7.5.3) to present the system to the user, and allow a range of commands to be sent. Most of the functionality is standard, but some parts were designed specially for In C.

Visually, the main customisation is the inclusion of the circular bars above each agent (see Figure 10.1). The operation of these is as follows:

- the upper bar reflects the position of the agent through the score relative to the mean of all the agents. This is not weighted by distance, so it gives a human controller an idea of which groups of agents are ahead or behind. As the agent moves ahead of the average, the bar extends clockwise, and becomes increasingly green; falling behind makes the bar grow redder and grow counterclockwise.

- the lower bar reflects the “pull” on the agent, as discussed above (Section 10.1.3). As the agent is more strongly pulled (by its close neighbours being ahead) this bar extends clockwise and becomes more green.

Several operations are available to the user from the contextual menu, but two of these are heavily targeted towards performing In C:

- The use of “Bump” messages is slightly idiosyncratic for In C; the idea is to have a relatively content free message — similar to a `bang` in Max/MSP, for example. This can be thought of as the equivalent of a nod, or glance. In general, there is no defined interpretation, and it is up to the agent to figure out how to respond. When playing In C, however, the bump is conventionalised, and interpreted as a request to move on to the next section of the piece. This provides a high level gesture that human participants can use to control the movement of the agents through the piece. This style of message could have a wider application; as an example it could be used when performing Canto Ostinato (see below) to suggest that this particular repeat is the last one. There are many circumstances where a single bit of communication can be effective.

- agents may be asked to take breaks and later rejoin the playing. Although this is a generic idea, there are only certain types of music where this would be appropriate. Again, this is a generally applicable idea, implemented in reaction to this particular score.

10.1.5 Example Performance

In order to give more of a feel for how the system works, here is a description of an archetypal performance using the system:

- the system is started, and a group of agents playing random instruments are arranged randomly around the space. None of the agents are playing, so there is no sound.

- One of the agents begins to play; proximal agents start to join in, and the sound becomes fuller.

- the user decides that this is too confusing, and moves most of the agents to the back of the space, so that only three agents at the front are clearly heard, while the rest produce a general background noise.

- the user bumps one of the agents in front so that they move onto the next section of the piece. After some time, the other two agents at the front move on to the new section.

- when a sparser section is encountered, the user decides that more instruments can be brought into play, and starts both moving agents from the back to the front, and creating new agents to join in.

- the piece continues, with the user adjusting the setup of the agent space according to current desires.

- eventually, to create an ending, the user "kills" the agents one by one, until there are none left, and the performance is over.

10.1.6 Discussion and Future Plans

Playing this piece uses several aspects of the agent system:

- the logical encoding of the score allows for complex behaviour to be specified concisely.

- the use of extramusical communication allows for the agents to quickly determine the actions of their peers, without complex musical analysis

- similarly, extramusical communication allows a human performer to take a role in shaping the output of the system, as the agents combine these directions with the score and their own "decisions" into musical output.

There are several directions in which the system could be improved:

- there are other directions in the score which could be encoded, to do with dynamics and texture. These are the features which a human performer can shape, so their absence is not a major problem, but alternatively this setup could be a testbed for a more detailed exploration of group dynamics in music.

- the agents currently play whatever section they are playing continually, and it would be more true to life if each agent could introduce short breaks, both to provide more space, and to allow exploring different offsets for the placement of the patterns.
- the sounds used by the system are produced by a General Midi sound module, and are not very exciting; also, the agents do not understand the sounds they are using. In terms of creating a higher quality performance, this would be an obvious place to start, although it would not help with investigation of agent behaviour.

It is hoped to arrange for a mixed human/agent performance of In C at some point in the future, where real human players are represented as agents in the system, and features of their playing extracted and used to influence the agents. This would provide a large scale demonstration of the Interactive paradigm from Section 4.

This case study has shown:

- that the system can function with a large group of agents, and create music in real time.
- that the score language can encode complex instructions.
- that the use of extramusical gestures allows for high level interaction with human participants, and reduced agent complexity.

10.2 Canto Ostinato

Canto Ostinato was written by Simeon ten Holt, in 1979. The score notes and some excerpts from the score are included in Appendix D, but a brief overview is given here. As the name implies, Canto relies heavily on the use of ostinato; the bulk of the piece is composed of repeated figures, based around equal five-note groupings. The structure of the piece is fixed, but musicians have decisions to make about how long to play each section for, and there is a range of material which can be used for each section. In contrast to In C, section changes must be shared between the players, but it is not specified whether section lengths should be determined in advance or during the performance.

There are several reasons why Canto is used here; the main points are:

- the score has a structure, and provides all the notes which are to be played, but leaves many possibilities open to the performers about how long to play each section for, and what collection of notes to use for a given section.

- the repetitive nature of the piece fits well with the types of analysis which have been implemented in MAMA. In particular, the original direction of the system towards jazz means that features were implemented to look at *grooves* and recurring patterns of accents.

- the piece is pleasant to listen to, and parts can be selected which are relatively easy to play, making it good for experimentation where participants will not have much time to practise, and will have to play the same piece many times.

- a piano duet was needed to provide a way to test the system and the theory it embodies when playing with humans.

More detail can be found in Chapter 11 about the use of the system under experimental conditions; the discussion here focuses on what capabilities of the system are explored by playing this piece, and what this shows about the system.

10.2.1 Encoding the score

Canto is scored for 2 to 4 pianists; in this case this was restricted to 2 to fit the experimental setup. Further, in order to keep experimental conditions consistent, a number of decisions have been made ahead of time, notably:

- each participant plays either the left- or right-hand stave from the central group of staves.

- the number of times each section is repeated is specified.

The score is then encoded using a set of `Sections`, each containing a `Channel` for the left- and right-hand parts, labelled so that an agent knows which hand to play. Some detail from the score has not been encoded:

- dynamic markings have been ignored, as this is an axis which was to be left entirely up to the performers.

- the score uses detailed indications to specify melodies built using the basic notes presented. These have been ignored, as it would be an extra effort to create a system which can both represent and interpret these, and this was not a necessary part of the experimental design.

An excerpt from the encoded score can be seen in Appendix D.

10.2.2 Using features

Once the notes are determined, what is left to the player is how the notes are played. This is one feature of Canto which makes it particularly appealing: as the phrases are repeated, the notes played fade into the background, and what becomes important is the changes in the way the notes are played. Melodies are picked out by accenting certain notes, and the performance is more akin to an improvisation with the constraint that only certain notes are available at certain times.

In order to do this, MAMA implements three types of feature, for the three aspects of playing notes available through MIDI; the onset, duration and "velocity" of notes are controlled, and for each of these an average value, a slope and a "pattern" is extracted - see Chapter 8 for more details. The conception here is that listening to the recordings of this piece shows the pianists using repeating patterns of accents or emphasis to create interlocking melodic/rhythmic structures, and that these structures can be captured by the features here.

Finally, Musical Actions are extracted from these features, and a deliberative module, trained on a corpus of human playing, is used to choose new actions to respond to the human input. The actions are then used to choose values that shape the musical output. This is fully explained in Chapter 9.

10.2.3 Discussion

At present, the system is only set up to play one half of a duet for this piece; the obvious direction to take would be to allow for a larger ensemble, and allow the full range of decisions allowed by the score. This would certainly make for interesting further work, and could lead to a full performance of the work, again using both human players and musical agents.

In contrast to In C, which looked at the capabilities of the Musical Middleware architecture to support extramusical communication, this demonstrates the capabilities of MAMA and the middleware architecture it implements to interact with humans using musical communication.

10.3 AgentBox

The AgentBox project explores the use of tangible interfaces to support multi-modal interaction between humans and musical agents. The initial impulse for the project came from a need for demos of informatics research in the Centre for Intelligent Systems and their Applications; the intended use is open days and science fairs, where a wide range of audiences should be able to quickly grasp the ideas involved in the demo. This project involved the creation of a physical artifact, a custom computer vision system and interfacing these to the existing agent system.

The initial conception of the demo was to create a space in which every person has a digital musical avatar. This avatar would track the person's movements in AgentSpace, and produce sound spatialised to appear from their location. The sound of the space would then reflect the activity within it, becoming more dense as more people entered. Each person would have a relationship with their own avatar, and indirectly with the avatars of others; groups of people would create islands of sound which would gradually synchronise; people whose avatars were playing clashing music would instinctively avoid each other.

Creating this kind of system is rather too involved for a demo, where it must be easy to set up under a variety of different conditions, so an alternative route was taken. Using a large cuboid, the AgentSpace discussed previously is reified, to create a tangible representation of the agent system. Coloured disks representing agents can be moved around, which alters the position of the agents, allowing people with no prior knowledge to interact with the agent system. Figure 10.2 shows the AgentBox being used.

10.3.1 Computer Vision component

The tangible interface for the AgentBox is built around a computer vision system; a camera is used to track the position of physical objects, which is then used to control events in the virtual world. An overview of the process is shown in Figure 10.3, and in more detail, the components are as follows:

- a large box with a translucent surface is the starting point for the system; this was built from scratch, using wood for the framework, and a 40% opal acrylic panel for the top surface. The lower surface of the acrylic was sanded so that light is transmitted in a diffuse manner; this means that only objects placed directly on the surface may be clearly discerned, and the camera sees anything above the

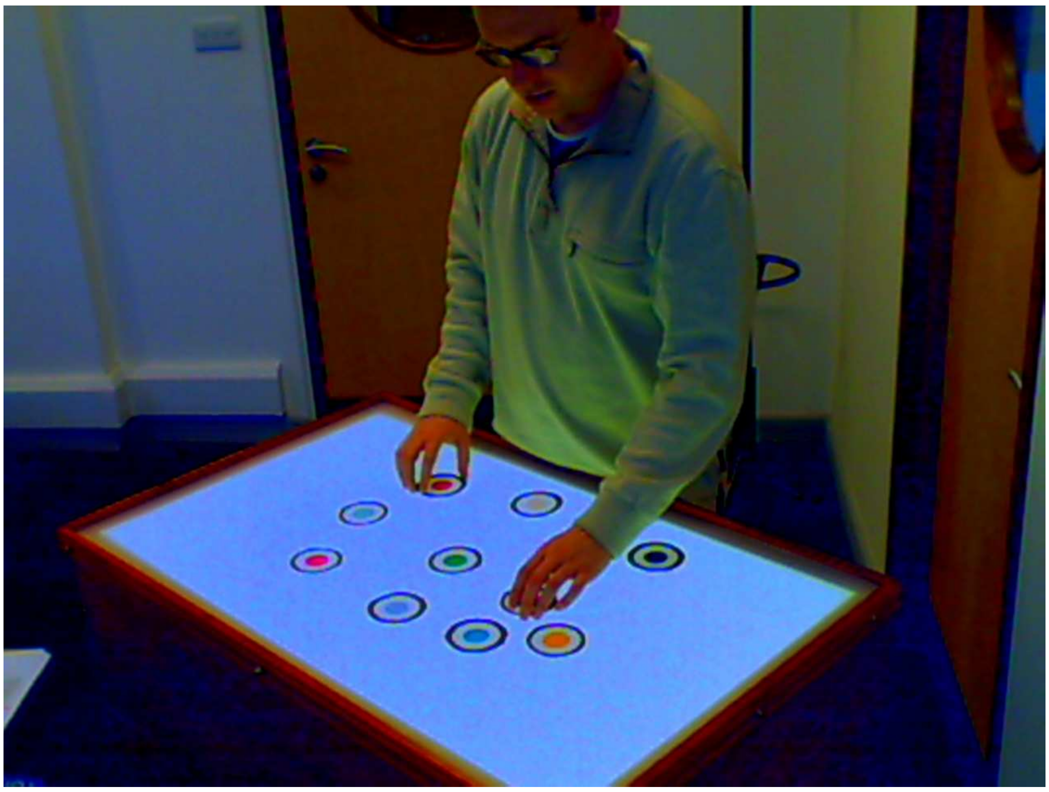


Figure 10.2: AgentBox in use

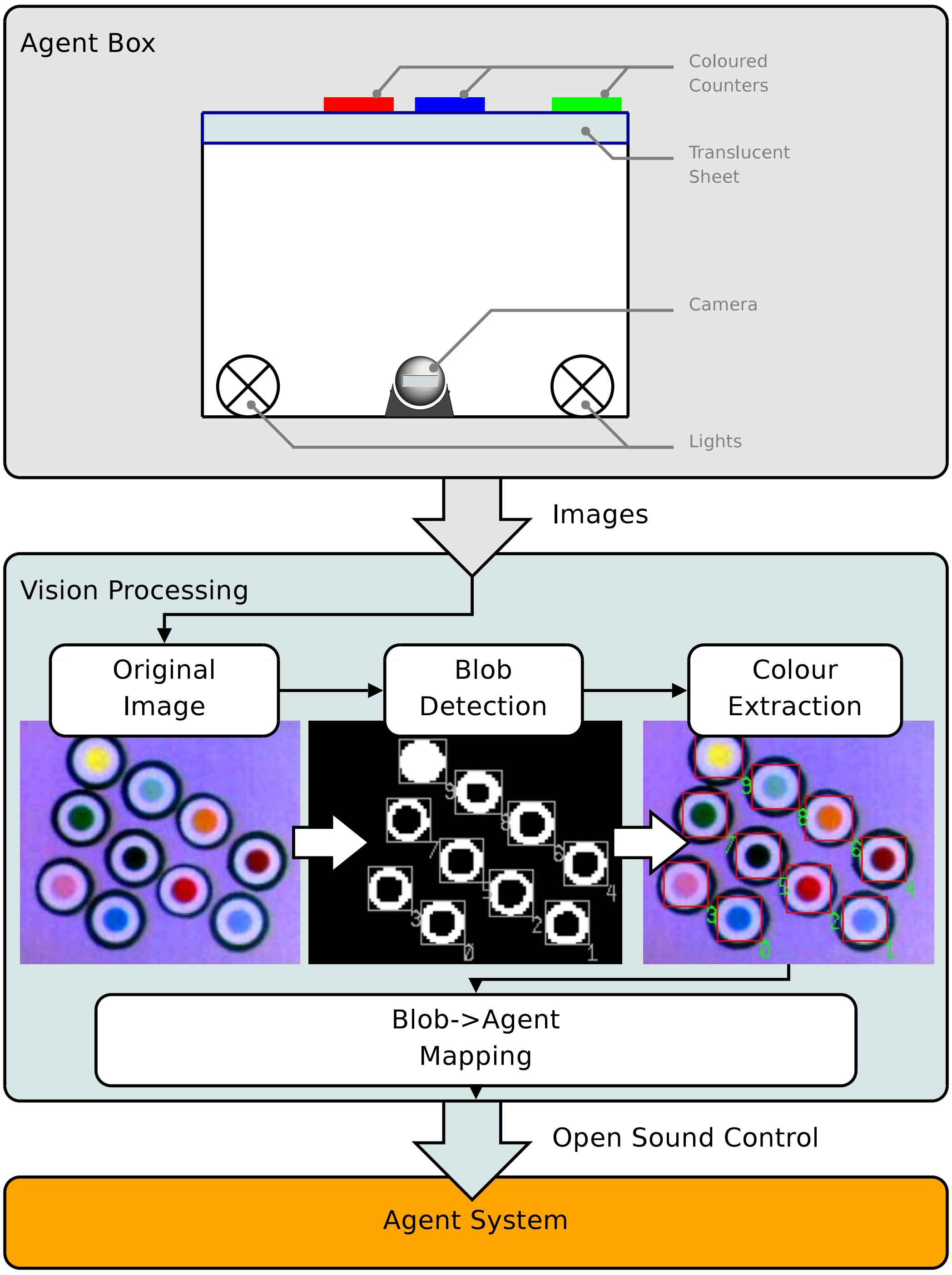


Figure 10.3: Overview of the computer vision system used in the AgentBox

surface as an indistinct blur.

- inside the box are two fluorescent tubes which provide light, and a webcam, pointed at the top surface of the box.
- coloured counters are placed on the top of the box; as they are in direct contact, they can be clearly seen by the webcam inside the box.
- the images produced by the webcam are analysed by a custom computer vision application, written in “Processing” [2], which works as follows:
  - the image is converted to monochrome, and passed into a blob detection algorithm, which extracts contiguous regions of light pixels. The counters are designed with a coloured circle in the centre, surrounded by a white ring, then a black ring. This ensures that the white ring of each counter will be detected as a separate blob.
  - the blob coordinates are then used to extract the coloured centres of each counter, which provides a set of (colour,position) tuples.
- these (colour,position) tuples are sent to the agent system using Open Sound Control [Wright, 2005].
- There are two modes of operation for the agent system:

  Discovery happens before the system as a whole is started. Here, the colours and positions of the blobs are shown, and any blobs which are too close in colour to be reliably identified are indicated. This allows the agent system to discover how many counters there are, and what their colours are. Once the user is satisfied that the correct set of counters can be detected, the agent system can be asked to start, and it will create the appropriate number of agents, and colour them accordingly in AgentSpace.

  Operation happens continually while the agent system is running. Each frame from the camera generates a collection of (colour,position) tuples, which are then matched to the colours of the agents which were set in the Discovery process. The colours are matched to the agent whose colour is most similar, up to a certain threshold; if the blob is not sufficiently similar to any known colour, it is ignored.

[2]http://www.processing.org

- once the (colour,position) tuples are matched to a particular agent, a custom agent sends position change request messages to those agents, along with some other possibilities which will be discussed later.

This provides a system which is capable of recognising approximately 12 different discs, running at approximately 5 frames per second.

10.3.2 Agent Capabilities

The AgentBox builds on the capabilities of the AgentSpace environment discussed previously (Section 7.5.3). Most of the same capabilities are offered, but in a multi-user, collaborative environment. The user's movements of the counters are interpreted as follows:

- the position of the counter on the Box relates to the position of the agent within AgentSpace. This has the "expected" effect that agents brought towards the front of the box become louder, while moving the agents left or right makes their output pan to the appropriate speaker.

- agents whose counters are removed from the Box are asked to stop playing. When the counter is replaced, the agent will start playing again, starting from the average section of the agents surrounding it.

- if a counter is "wiggled", this sends a "Bump" message to the agent, as discussed in the section on In C.

Although this is a relatively simple set of possible actions, it quickly becomes clear that there are many possibilities with this space; users can:

- rapidly shape the texture and timbrality of the piece, by shuffling the placement of the counters

- create situations akin to solos and duets by bringing small groups of agents forwards and pushing the rest backwards, ensuring that the soloists are both loudest, and paying most attention to each other

- use musical output as a basis for arranging musicians spatially; for example, a line of musicians playing the same section arranged from front to back will give a sense of temporally and spatially displaced echoes.

These capabilities are latent in the agent system, and can be explored using a traditional mouse and keyboard interface, but this is a significant “barrier to enjoyment” for many people; it is fiddly, and it is only possible to manipulate a single agent at any given time. Adding a tactile, natural system greatly increases both the initial and lasting excitement of people using the system.

10.3.3 Discussion

The contribution of the AgentBox project to the overall thrust of the work is mostly in exploring the way that the system can be interfaced to the rest of the world, and used by the general public. The main points to note are:

- the relative ease with which the system could be integrated into the agent paradigm demonstrates the power of the agent system approach, and that it is functioning as musical middleware. The entire AgentBox project was completed in a (busy) three weeks, and most of that time was spent on creating a robust physical and software vision system. Communication between the agent system and the AgentBox is managed by a single agent, which receives OSC messages and injects them into the agent system.

- the system allows for intuitive, high level control of the agent’s playing, using extra-musical communication. This allows non-musicians to shape the performance, and gives a sense of “performance” or “liveness” which is easily accessible. This is aided by the physicality of the counters which represent the agents.

- This is an instantiation of the “Installation” paradigm, (Section 4) in that the system will keep playing music autonomously, but participants have ways in which they can alter the music which is being produced.

There are several things which could be improved about the setup: spatialisation is not very strong, a wider range of interactive gestures could be implemented, and it would improve the sense of physicality if the agent space was projected onto the same surface the counters are resting on. However, these would require extra work, and do not detract from the value of the present system as a proof-of-concept. Also, the system is put together using very cheap commodity hardware — the accuracy and speed of tracking and identification could be improved significantly by using a more advanced camera, and the feedback to the user could be greatly enhanced by the addition of a projector inside the box displaying information from agent space on the translucent top surface (e.g. Jordà et al. [2005]). Finally, a more developed computer vision system could be used, e.g. recactiVision (as used by the reactTable, again Jordà et al. [2005]).

In terms of interactivity, there are two kinds of effects:

- moving agents around and removing or adding them create immediate responses, with simple, direct mappings which can easily be understood and explained,
- manipulating the distances between the agents affects the behaviour in a much less obvious manner, creating subtle relations between the musical output of neighbourhoods of agents.

It is not always immediately clear to participants how the subtler mappings work, and especially when there are a large number of agents creating sound, it can be difficult to tell what effects one's manipulations are having on the sound produced. This level of indirect manipulation requires people to invest an amount of time into understanding the effects of their actions — it pushes the interaction from being one of control and reaction into being an influencing of a largely autonomous system, with complex reactions to simple events.

Finally, the visceral enjoyment of being able to manipulate a large number of agents directly using hands cannot be overstated. My personal use of the system quickly led to creating spatial patterns of agents which passed musical phrases between each other; continually shifting timbral textures by rotating all the agents at once with both hands; solo style situations where most of the agents were pushed to the back to provide a background susurration while a changing group of agents were brought to the front to provide a focal point.

10.4 Conclusion

These case studies have shown several abilities of the Musical Middleware architecture, and the MAMA system:

- the architecture is implementable, and it allows for new capabilities to be added relatively easily, and to be *loosely coupled*[3] with the core system.
- the infrastructure can handle a group of musical agents interacting with humans.
- the system can use both musical and non-musical communication when interacting with humans

[3]i.e. there is a simple interface with no complex dependencies.

Finally, the system has implemented several of the paradigms set out in Section 4, in particular Performance, Interactive and Installation, and played two quite different pieces of music.

Chapter 11

Experimental Design, Implementation and Results

This chapter discusses an experimental hypothesis about the performance of computational musical agents, an experimental design to test it, and the results of running a pilot for the experiment.

In order to perform an experiment which could produce a generally useful result about the cognitive processes involved in making music, a great deal more work would need to be carried out at several levels. This experiment aims at different goals:

- to provide a testbed to make sure that the system functions as required in real life situations, both at an operational level, and at a theoretical level.
- to design an experimental setup which can be used to measure a notion of interactivity for computational musical agents, and to explore the issues involved in carrying out such an experiment.

11.1 Overview and Hypotheses

The hypothesis H_1 of this experiment is:

> H_1: the addition of an understanding of Musical Act Theory (MAT) to a computational musical agent increases the quality of interaction with human musicians

This hypothesis was tested by creating a system which can perform a piano duet with a human participant, informed by MAT, and analysing the responses of the participant to the system and several baselines. This allowed the testing of a series of more concrete hypotheses (see Section 11.2.2), which together will provide support for the main hypothesis.

This experiment is not concerned with whether the music produced in each interaction is “good” - the actual quality of the music is of no direct importance here. What is under test is the relationship between the musicians (human or computational) during the interaction. The main reason for this is that the Musical Act Theory is directed towards the communication which occurs when people play music together — it does not address the music created as a result of this communication.

In overview (see Figure 11.2 for a system diagram):

- participants are sent a score to familiarise themselves with, and brief instructions covering what they will be asked to do.
- participants enter the room in pairs, and sit at individual keyboards, wearing headphones and separated by a screen.
- each participant answers a few demographic questions concerning their playing capabilities, familiarity with this style of music etc.
- the participants are asked to play a series of excerpts from the score, answering a battery of questions after each excerpt directed towards the quality of interaction with their partner.
- in each excerpt, the participants may be playing together, or each may be playing with an instance of the system in a variety of configurations, or they may be playing with recordings of previous participants.
- factor analysis is performed on the questionnaire results to recover values for variables relating to interactivity which are used to test the main hypothesis by means of several subsidiary hypotheses.

11.1.1 Generation of main hypothesis

The main thrust of Musical Act Theory is a framework to model the interactions which take place when people play music together. It is not designed with any particular style or type of music in mind, although inspiration is drawn from more heavily improvised musics. The embodiment of MAT in a computational system should allow the computer to interact in an informed manner with human musicians, through the medium of music.

In order to verify this, a comparison must be made between a system which uses MAT to inform its musical decisions, and one which does not. Since the area we are interested in is interactivity, a technique must be developed to test interactivity, and the comparison must be solely between the interactivity scores of the two conditions. This led to the main hypothesis of this chapter:

> H_1: the addition of an understanding of Musical Act Theory to a computational musical agent increases the quality of interaction with human musicians

11.1.2 Definition of terms

For the rest of this chapter, the discussion centres around two terms, relating to the features used to evaluate one's musical partner. The objects of discussion are believable musical agents - that is systems which can function as an equal partner in an improvisation with human musicians. The two features for analysis are:

Interactivity is concerned with the manner in which the actions of one participant influence those of others. One player might match the timing or harmonic output of another; the use of certain material by one player may push the other participants to use contrasting material. As a measurement, participant A's score for interactivity measures how participant B responds to A's actions: does A's output affect B's, and if so, in what ways? Is it a simple copying or fitting in? Do B's choices surprise A, or push the interaction in new directions? Is B predicting where A is heading, and acting accordingly?

Expressiveness is used in the sense of "human sounding"; that is, on a superficial level, does the playing sound as if performed by a human musician? For instance, many pieces of software have a "humanize" function, where noise is added to the onset and volume of notes makes the recording easier on the ear. Also, much research on expressive performance focuses mainly on the relation between a single performer and the score (e.g. Widmer and Goebl [2004])

There is clearly an overlap between expressivity and interactivity - a musician's playing is likely to be considered more expressive if it fits in with the playing of the rest of the group. However, a system might be:

Expressive but not interactive A system designed to play scores expressively might not have any facility for accepting user input, or might be limited in scope (e.g.

simply following a soloist). Music Plus One (MPO) [Raphael, 2001] would be a prime example — it fits recordings of the accompaniment to the output of a soloist.

Interactive but not expressive A system whose output is highly influenced by the playing of others around it, but which either has no expressive musical features and simply renders notes in exact metrical time at constant velocity, or which has a non-human form of expressiveness, that seems random or jarring to human ears. Early versions of Swarm Music [Blackwell, 2003] might be placed into this category — it is clearly highly interactive, but some of the music lacks expressive features.

In order for a musical agent to be believable and interesting to play with, both of these qualities are desirable. The reason for exploring and emphasising this distinction is that the central hypothesis concerns the interactivity of the system, and this may be obscured by issues of expressiveness; as an example, the output of the human and the MPO system described above might be reasonably expressive, yet there is no deep interaction going on.

Three terms from Section 4.2.5 are used heavily in this chapter:

Analysis is the extraction of high level features from the musical surface

Generation is the creation of music from high level plans, and may include a limited computational load, but no serious reasoning

Deliberation is the formation of high level plans, based on the input from others and any goals which the agent may have

11.1.3 Chapter Structure

The rest of this chapter is concerned with the following tasks:

- creating an experimental design which tests the main hypothesis assuming that a method of measuring interactivity is available.
- creating a method for measuring interactivity.
- implementing an experiment which carries out these measurements.
- analysing the results from this experiment.

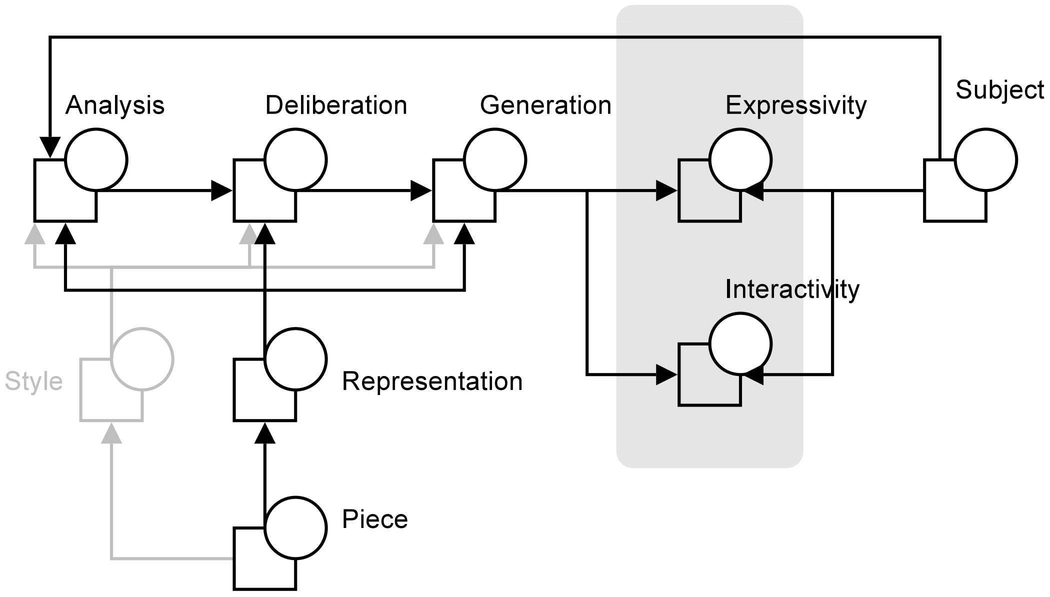


Figure 11.1: Variables involved in the measurement of system performance

11.2 Experiment Design

This section explains the experimental factors used in the experimental design, and creates a set of testable hypotheses directed towards supporting the main hypothesis. It is assumed that a measurement technique is available which allows the measurement of expressiveness (E) and interactivity (I) — these measures will be developed in Section 11.2.3.2. The discussion of the experiments in this section are constructed from the point of view of a single human participant playing music with one other agent. Even though this other agent may also be a human participant, only one side of the experiment will be discussed at any given time.

11.2.1 Experimental Variables

Diagram 11.1 gives a causal model for the performance of the system in the experiment, playing with a human subject. This is derived as follows:

- Two main factors are postulated, Expressiveness and Interactivity, as defined above. For a given feature, these depend on the output of the agent, and the subject which it is playing with.

- The output of the agent ultimately emerges from its generative subsystem, hence this is the only direct causal link to the Expressiveness and Interactivity factors.

- The generative system of the agent is influenced by its deliberation, which is in turn influenced by the analysis system, working over the output of the subject.
- The agent's representation of the piece will guide its analysis, generation and deliberation; this representation is dependent on the piece chosen[1].
- Finally, the style which the piece is to be played in, combined with the stylistic knowledge of the agent influences the analysis, generation and deliberation of the agent. However, since this is being held constant through the experiment, it is only included for completeness.

In order to support our hypothesis, we must demonstrate that the interactivity score for the system is higher when it is using its deliberative mechanisms; this means that we must construct a situation where all of the variables are the same except for the deliberation section. Our two main conditions are hence:

Deliberative is the system running with all of its components intact; its deliberative mechanisms are used to construct high level plans in response to the input of the subject. The interactiveness and expressiveness scores are written as I_{Delib} and E_{Delib}.

Mirroring "short circuits" the deliberative process; all of the features produced by the analysis module are fed directly to the generative module, so that the agent copies as closely as possible the output of the subject. The scores are written as I_{Mirror} and E_{Mirror}

These cover our main hypothesis, which can now be written as

$$H_{1^*} : I_{Delib} > I_{Mirror}$$

11.2.2 Supporting Hypotheses

In order for the previous inequality to be a good substitution for the original hypothesis, it is necessary to show that:

- I and E are measures which relate to the natural concepts of expressiveness and interactivity as defined at the start of this chapter

[1]representation and piece are not the same variable, however, as one piece may have multiple representations

- This experiment provides sufficient sensitivity to differences in the variables, and the values produced are free from floor and ceiling effects.

To do this, several benchmark conditions are introduced, along with expectations about their behaviour:

Human is the condition where the subject is playing with another human. In this case, this is the "gold standard" for both the E and I scores. While it may be theoretically possible to have a system which is better in one or both of these scores, it is considered extremely unlikely. It is therefore expected that E_{Human} and I_{Human} are the maximum for their respective scores.

Recording is the condition where the subject is playing with a recording of a previous human participant. In this case, E should be high - as it is real human playing, but I should be low, as the recording is not being influenced at all by the subject

Straight is the condition where the human is playing with the system, and the system is outputting all of the notes in the score with no expression - every note is constant volume, precisely timed and a length exactly proportional to that in the score. This condition is expected to have the lowest values for both E and I, as no expressive or interactive work is undertaken.

The supporting hypotheses this gives rise to are:

H_{human}: E_{Human} and I_{Human} are highest of all the scores.

$H_{straight}$: $E_{Straight}$ and $I_{Straight}$ are the lowest of all the measurements; since they represent a completely mechanical rendering of the score, there should be no interactivity, and no expressiveness.

H_{record}: $E_{Record} > E_{Straight}$, but $I_{Record} < I_{Human}$. Playing with a recording of a human should have a higher expressiveness than playing with a mechanical rendering of the score, but it should have a lower interactivity than playing with a live human

H_{anal}: $I_{Straight} < I_{Mirror} < I_{Human}$ When the system analyses the human's playing and mimics it, this improves interactivity, but not to the level of a human player.

There are other possible hypotheses, but these are the ones necessary to support H_{1^*} as a valid recasting of H_1.

	E	I
Human	large	large
Straight	small	small
Recorded	large	small
Mirror	medium	medium
Deliberative	medium	large

Table 11.1: Ideal Results

A set of ideal variable values which demonstrates all the necessary characteristics is shown in Table 11.1[2].

11.2.3 Experimental Stages

If time and resources were no object, the experiment would be conducted in two phases, as outlined below. This was not the case, and the experiment was actually run as described in Section 11.2.3.2.

11.2.3.1 Phase 1: Design Validation

The first phase of the experiment is used to validate the design. H_{human}, $H_{straight}$, H_{anal} and H_{record} are confirmed on a run with a limited number of participants. Some possible failure modes are:

$E_{Mirror} \not> E_{Straight}$ If the mirroring system fails to produce a significant improvement in expressivity over a basic rendering, then it is poor at introducing the kinds of expressive features which humans are sensitive to, or the measure of E is not functioning correctly. This would not necessarily pose a large problem for the validity of the experiment, as it does not affect the measurement of interactivity.

$I_{Mirror} \approx I_{Human}$ If the mirroring system is perceived as being highly interactive, it would reduce the chance of finding a significant difference between I_{Mirror} and I_{Delib}. In the extreme case, if it scores as highly as a human player, the experiment cannot show that the deliberative system is better, and it indicates a ceiling effect in the experiment.

[2]It should be noted that while these values have all the necessary qualities not all of the relations derivable from the table are necessary predictions.

$I_{Record} \approx I_{Human}$ If the interactivity when playing with a recording is close to that when playing with a human, then the score is not a valid measure of interactivity, and an experimental redesign would be required.

$I_{Record} \approx I_{Mirror}$ If the interactivity when playing with a recording is close to that when playing with a mirroring system, then some doubt is cast on the validity of the interactivity score. However, this might be because the output of the mirroring system scores poorly for interactivity, and hence less of a serious problem for the measurement of of the quality of the deliberative system.

11.2.3.2 Phase 2: Main Experiment

Once the validity of the above hypotheses is clear, investigation of H_{1^*} is possible, given that:

- expressiveness and interactivity are measured in a meaningful manner
- it is possible for our results to support H_{1^*}
- we are avoiding any floor and ceiling effects in the design of the experiment

Subjects are now run in the deliberative and mirroring conditions, with human, recording and straight conditions added as a safeguard to ensure that the previous investigation still holds true.

At this point, it is possible that the results may or may not support H_{1^*}, and this translates into supporting (or failing to support) H_1.

11.3 Experiment Implementation

The previous section provided a high level view of what the experiment will show. This section details the mechanics underlying the claim, especially the development of measurements for E and I.

In overview, participants play piano duets, and rate the quality of interaction using a questionnaire. Without the participant's knowledge, sometimes these duets are with another human, and sometimes they are in a variety of artificial conditions. The questionnaire scores are then used to determine the quality of the interactions. In overview:

- participants are run in all five conditions.

- the questionnaire results from the Human, Straight and Record conditions are used to construct a factor loading matrix from questions onto E and I.
- this matrix is used on the data for the Mirror and Deliberative conditions, to construct E and I scores which can be compared.

11.3.1 Experimental Setup

In order to maximise psychological validity, it is necessary to make the conditions as close as possible in all aspects except the variables which are being purposefully manipulated. In order to keep personal bias out of the experiment, it is desirable that participants keep their belief that they are playing with another human. To ensure this, participants are run in pairs, so they believe they are playing with each other. Although the questionnaire makes mention of the possibility that they are playing with recordings, it is never suggested that they may be playing with a computer system.

There are several ways in which the system at present is clearly distinguishable from a human, which must be controlled for:

Physical presence and perception When humans play together, they will look at each other, and a variety of information can be exchanged. Similarly, they may discuss things verbally, or count out loud through difficult sections. In order to reduce this behaviour, a screen is placed between the participants so that they cannot see each other.

Playing different parts of the piece Each person is playing with the system in some configuration, and they are likely to be playing different parts, in different manners to each other. In order to prevent this from being obvious, headphones are worn, so that the participants may only hear each other in the condition where they are playing together. Also, the conditions are arranged so that the participants are playing complementary parts at any given time[3]

Expressive use of tempo Since the system is not currently set up to follow tempo or perform beat tracking, all conditions must be played against a metronome, produced by the system.

[3]for example, even though participants A and B are playing with virtual partners, participant A is playing part X for section N and participant B is playing part Y for section N, which are the same parts they would be playing if they were both playing section N together

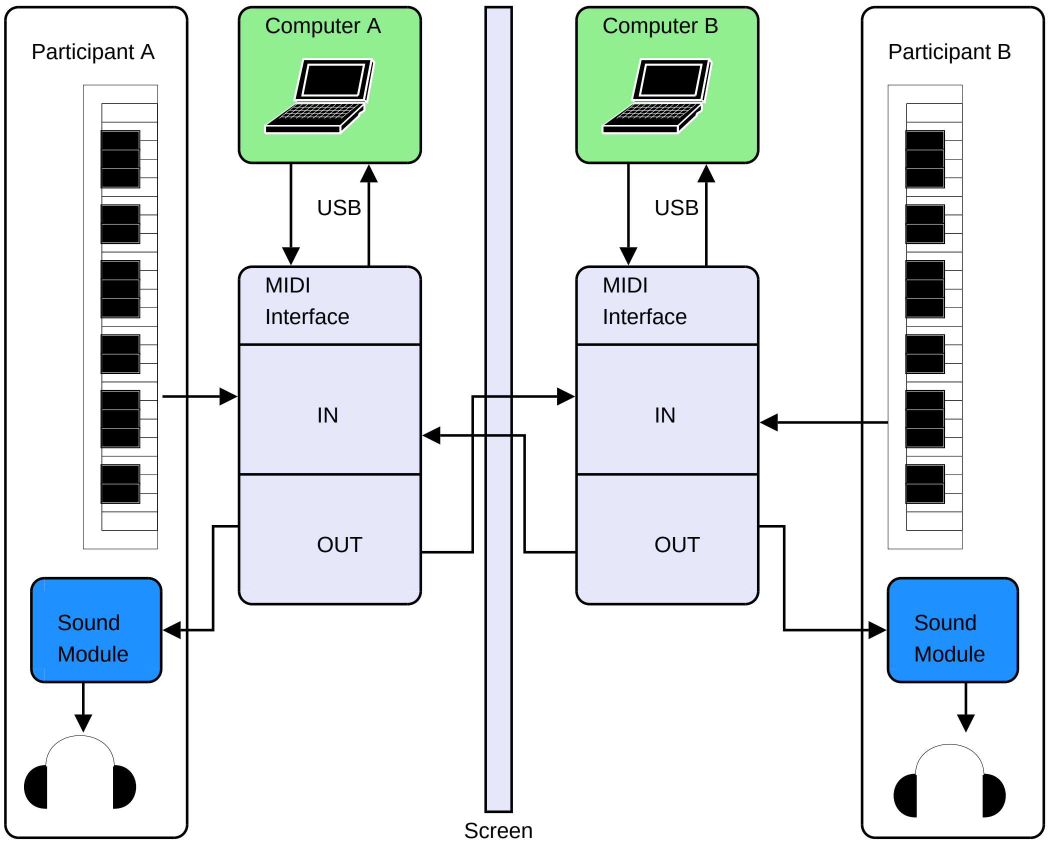


Figure 11.2: Experimental setup block diagram

The setup is shown diagrammatically in Figure 11.2. The playing of each participant is sent to a computer, which sends

- that participant's playing,
- the systems's playing,
- a metronome.

to a sound module connected to the player's headphones. In the condition where the two participants are playing together, one computer provides the metronome, and distributes the output of the two players to each other. Each computer also records (as a MIDI file) the complete output of every interaction it runs.

11.3.2 Questionnaire Design

The questionnaire is directed toward measuring two quantities - expressiveness and interactivity - which are subjective measurements of a participant of the interaction they have been engaged in. A complete questionnaire can be found in Appendix E. The questionnaire consists of three parts:

- a preface describes the experiment, and asks some questions of the participant, covering age, musical and pianistic competence, familiarity with the style of music being played and how much they play with other people.
- the main body of the questionnaire
- a coda, which asks questions about the participant's personal performance in the experimental setup compared to their normal performance, and queries any difficulties or suggestions relating to the setup.

Questions are designed to probe the participants opinions about certain subjects. As such, they are presented as Osgood Semantic Differentials [Snider and Osgood, 1969] — a pair of bipolar possibilities in response to a particular question, with the participant being asked to choose between seven different positions between the two extremes, e.g.:

How difficult did you find it to play with a partner you could not see?

very hard __ __ __ __ __ __ __ very easy

The other standard choice here would be a Likert scale [Likert, 1932], but it was felt that the use of two extremes in a semantic differential more accurately captures the nature of the questions which were asked, and has less chance of biasing the results.

The main body of the questionnaire is arranged around a series of excerpts the participant is asked to play (see next section for details of the excerpts). Each of these excerpts is played in a different condition, although the participant is not aware of this. The order of the excerpts and conditions, and the order of questions after each excerpt is randomised. The participants play each excerpt, and then answer questions on it before playing the next.

The questions in the questionnaire are directed toward exposing the two quantities of interest (expressiveness and interactivity), and can be seen in short form in Figure 11.3. Since the true relevance of each question to these quantities is not known, factor analysis is performed to recover factor weightings for the questions, and values for the quantities.

11.3.3 Choice of piece

The choice of piece to be played is an important factor in this experiment. The current sophistication of the system does not extend to generating new notes, so a piece is needed which allows for a wide range of choices and expression, while also supplying the notes to play. For this reason, "Canto Ostinato" by Simeon ten Holt is used. This piece may be played by between 2 and 4 pianists, and is highly tonal, makes use of structural harmony and cause and effect (tension-release)[4]. It is written as a single score, with a selection of staves containing variation which the players can choose to play from. As the name suggests, the piece is made up of repeated ostinato phrases. These are divided into sections, with each section being repeated indefinitely. The large scale decisions available to players are:

- how many times to play each section

[4]paraphrased from the score notes in Appendix D.

Interactivity Questions

1	”Did you feel the other participant was paying attention to your playing?”
2	”Did your partner adopt your ideas in their own playing?”
3	”Did the other participant surprise you with their playing?”
4	”Did your partner play their part in a way you had not experienced before?”
5	”Did your partner introduce new ideas?”

Expressivity Questions

6	”Did your partner articulate phrases expressively?”
7	”Did you and your partner stay in time?”
8	”Did your partner vary their timing in a musical manner?”
9	”Did your partner use dynamics expressively?”

General Questions

10	”Did you enjoy playing the excerpt you just played?”
11	”Would you choose to play with your partner again based on the excerpt you just played?”
12	”How would you rate your partner’s competence based on the excerpt you just played?”
13	”Was your partner playing live, or was it a recording of another person?”

Figure 11.3: Questions used in questionnaire to probe I and E

- which variation/stave(s) to play
- when to play, and when to be silent
- certain sections allow movement backwards as well as forwards in the score.

On a smaller scale, the effect of the constantly repeated phrases is that the phrases themselves become background, and the ear perceives the differences between them. As a result, much of the way the piece sounds is dependent on the timing, dynamics and articulation used by the performers. Stationary patterns of accents, timing and articulation are used to pick out certain notes, to create structure and melody from the constant stream of notes.

The temporal nature of the piece makes it amenable to excerpts being taken; the use of repetition to establish "time [as] the space in which the musical object floats" allows for a dialogue to build up around small structural units, with progressive refinements before moving on to the next section. Three short excerpts from the piece have been chosen. This choice is intended to allow exploration of aspects of the piece, but not require the participants to spend an undue amount of time familiarising themselves with the piece. The excerpts are:

Sections 2-9 are the beginning of the piece, and relatively static; this allows the participants a lot of freedom to work together to explore dynamics and timing, on a stable harmonic base

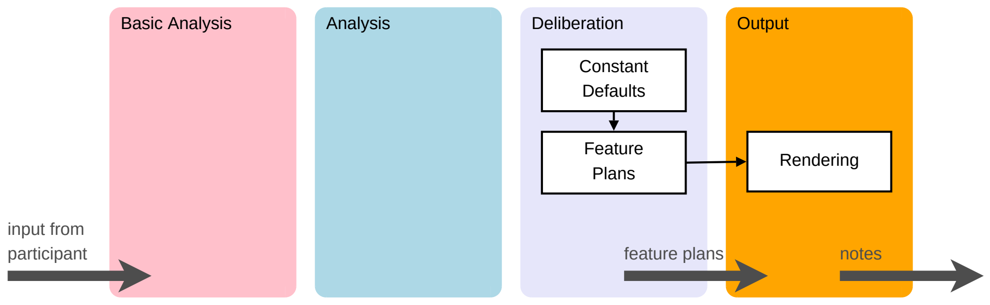
Sections 69-78 are more "expressive" than other sections, with some longer sections containing more defined melodies, and faster harmonic changes

Section 88 is the bridge, and contains many similar sections which the performers may freely move backwards and forwards in.

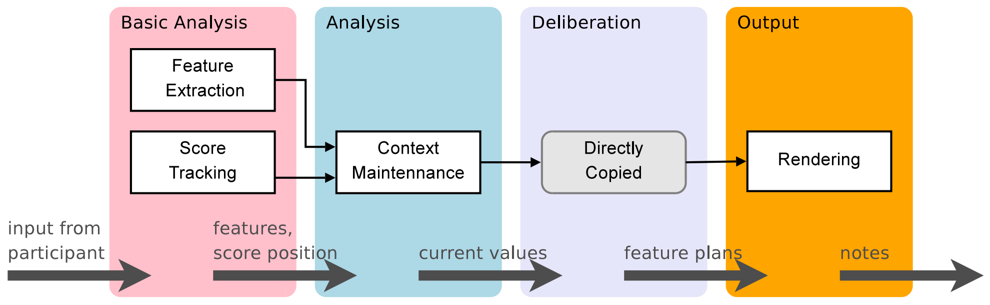
For each excerpt, the participant is told which sections to play. In order to reduce the workload for each participant, they are asked to only play a single line from the piece - either the left or right hand from the central stave.

11.3.4 Capabilities of the system in relation to the piece

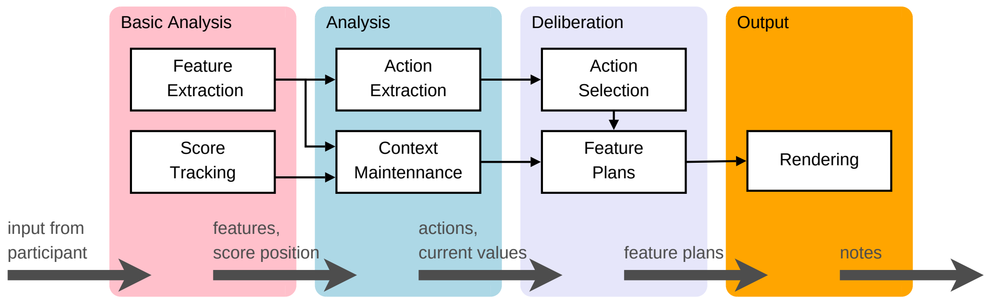
As noted previously, the piece allows for a range of structural and featural decisions to be made by performers. In a performance situation, some of these would be determined in rehearsals, and some may be left open to spontaneity during the performance itself;



(a) Straight reasoning configuration block diagram



(b) Mirroring reasoning configuration block diagram



(c) Deliberative reasoning configuration block diagram

Figure 11.4: Comparison of different system configurations

this would be dependent on the preferences of the performers and their chosen style of interpretation.

This allows for the complexity of decision-making to be tailored to the capabilities of the system at the time of experimentation - any decisions which the system is not capable of making can be taken as predetermined, and passed onto the participant.

At present, the functioning of the system covers what would typically be called *expressive* features - no decisions are made about which notes to play, simply about *how* to play them. However, in contrast to typical notions of expressive performance, the features are analysed relative to the features expressed by the other participant, without making use of information from the score or a knowledge of music theory. Hence, even though the features used would normally come under the heading of expressive performance, they are decided in an interactive manner, and are hence treated as interactive.

The system deals with note onset times, note lengths (relative to their scored values) and note volumes. For each of these features, the system works with the average value, the rate of change of the values, and the pattern of residuals left once the underlying trends have been taken into account. For more details on this, see Section 8.2. Decisions about how long to play each section for, and which part to play are assumed to have already been made, and are encoded into the score and specification given to the system at the beginning of each excerpt.

The system is required to provide three different modes of music generation: straight, mirroring and deliberative. This is provided in the following manner: (see Figure 11.4 for a diagrammatic overview)

Straight uses the `StraightReasoner` class, which totally ignores all input, and produces rendering plans which maintain a constant volume and proportional note length, and play every note exactly at the metrical position indicated by the score.

Mirror uses the `ValueReasoner` from Section 8.4.1, which extracts the features noted above from the playing of the participant, and constructs render plans which embody these features - mirroring the expressive output of the person it is playing with.

Deliberative uses the `SequenceReasoner`, which uses the Musical Act Theory developed in Chapter 6 and the deliberation method from Chapter 9, trained on a corpus of data from human-human interactions, to construct responses. Chapter 9 gives full details; the important point is that this is the setup which uses Musical Act Theory as the basis for constructing its responses.

The two further conditions are:

Player is when the two participants are playing together

Record is when each participant is playing with a recording of a previous human performance. For consistency, the same recordings are used for each trial; these recordings were produced using the same setup as the main experiment, but only running people in the Straight, Mirror and Human conditions.

11.4 Analysis of Results

This section describes the investigation of the properties E and I with respect to the raw question data, the extraction of appropriate factors from the raw user question data, and the comparison of these properties between different conditions.

The main dataset for the experiment consists of the questionnaires filled out by each participant. Each participant was asked to play a series of excerpts, and rate the performance of their partner using a set of questions after each excerpt. Each of these sets of questions is the result of an interaction, so their answers will be termed an *Interaction Result*, or IR. This questionnaire also contains a series of pre- and post-questions, assessing the participant's general attitudes and competences for making music and specific questions about their overall performance on the day, which have not been included in this analysis[5].

The data for each IR can be represented as a tuple of integer value responses to the questions:

$$IR = (q_1, \ldots, q_n)$$

Each IR was also played by a person, playing a one hand for a section of the piece,

[5]they were intended to allow extra methods of analysis, such as comparing quantifiable skill levels (e.g. amount of practice, years spent playing) and perception of personal performance on the day with the ratings given by participants in the main experiment. In the end this level of analysis was not considered appropriate

against a partner (condition). So a complete *Interaction* can be represented as[6]:

$$\begin{aligned} In &= (p,h,r,c,(q_1 \ldots q_n)) \\ p &\in People \\ h &\in \{Left, Right\} \\ r &\in \{1,69,88\} \\ c &\in \{Player, Recording, Straight, Mirror, Reason\} \end{aligned}$$

From this the procedure is as follows:

- preprocessing the data so that we can separate the data by condition.
- Separate out the three "baseline" conditions - Partner, Recording and Straight and perform a factor analysis on all of the IR's in these conditions, to recover factor loadings for two main factors
- use the characterisations of the different conditions to decide which of the factors represents E and which is I, by looking at the factor loadings for the different groups of questions.
- separate out the two conditions under test - Mirror and Reasoning - and use the computed factor loadings to generate E and I scores for each participant in each condition.
- perform a paired t-test on the I samples between these two conditions.

11.4.1 Data capture and participants

The dataset was collected from keyboard students at Napier University, all of whom had been playing piano for more than five years, and considered it their main instrument.

There were some issues which should be noted at the beginning of the analysis:

- the data set is very small, with only five participants having been run. When running experiments involving skilled people of any variety, with limited resources, it can be difficult to find an appropriate number of participants. If the effect size is estimated as being "large", then 28 participants are needed to have a power of

[6]r refers to the sections of the piece chosen as excerpts

0.8 Cohen [1988, page 102]. Several attempts were made to work with pianists from the University, but due to the difficulty of combining pairs of pianists in a room with the correct equipment, this proved unworkable. Eventually, in order to find close to this number of skilled pianists an arrangement was made with a local keyboard school to work with the students there. This was expected to yield 16 participants, which while not as many as desired would nevertheless have a reasonable chance of showing a result: with $\alpha = 0.05, d = 0.8, n = 16$ the power is 0.76. In the end, due to several factors, only five pianists could be run. While this would be an issue in an experiment which depended on conclusive results, for the pilot study here, it is a reasonable starting point.

- the experiment asks participants to familiarise themselves with the piece before the experiment. Unfortunately, not all of the pianists did, although they were of a sufficient standard that they quickly adapted. Due to the fact that this level of familiarity was not reliably classified, and the fact that the data set is already small, runs from these pianists are included in the analysis.
- the structure of the piece proved difficult for several of the pianists; in many interactions, human players reached the end of their score well in advance or well after the system. Anecdotally, it seemed to be that the humans were more likely to finish early than late. This indicates that counting the number of repeats (in an already repetitive piece) caused significant difficulty.
- due to the absence of one participant, their prospective partner performed the experiment alone. This had two effects: they were aware that they were not playing with a human, and all of the “Player” conditions had to be skipped. Again, due to the already small data set these results were used in the analysis.
- there was an issue with the module used in the Deliberative condition, which is discussed in Sections 11.5.2 and 9.4.2. The effect of this bug was that the Deliberative condition was behaving less deliberatively and more like the Mirroring condition than was expected.

11.4.2 Initial data preprocessing

Each participant performed 12 interactions[7], the conditions for which were randomly chosen. This means that each person will have played a different number of excerpts

[7]except where this was impossible, as noted in Section 11.4.1

		Participant		
		p_1	$\dots$	p_n
Condition	c_1	IR_{p_1,c_1}	$\dots$	IR_{p_n,c_1}
	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	c_n	IR_{p_1,c_n}	$\dots$	IR_{p_n,c_n}

Table 11.2: Preprocessed data set: Interaction Results by Participant and Condition

in each condition, and that there will be a different number of interactions in each condition overall; the data for each person in each condition is hence averaged. At the same time, data relating to which hand and excerpt is played is ignored. The dataset hence becomes a 2d matrix, with participants on one axis and conditions on the other, with each cell containing the averaged list of question responses from that person in that condition - see Table 11.2.

As well as discarding the hand and excerpt data, it should be noted that the data about which participants were playing together has been implicitly discarded. This is an assumption which needs to be stated, as it could potentially have a large effect on the perceived quality of the “Player vs. Player” interactions.

11.4.3 Direct analysis of question data

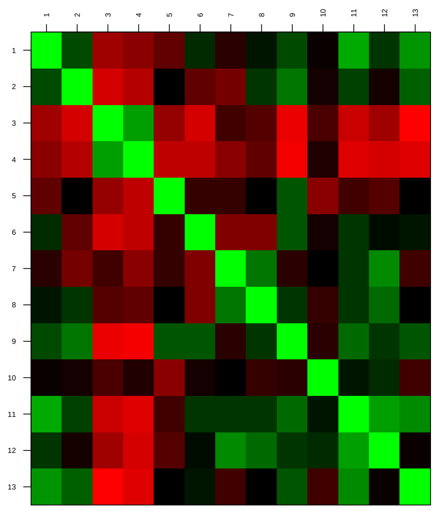
Before the detailing the main part of the analysis, an exploration of the raw question data is described. Here, the question answers by participant and condition are used (see Table 11.2). In order to get a general feeling for the effectiveness of the questions, two versions of the covariance matrix are shown in Figure 11.5.

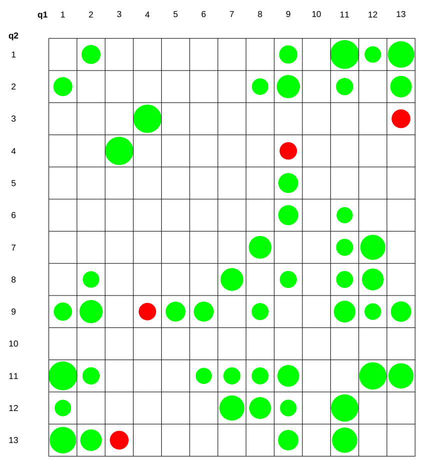
From these plots we can see that:

- in general, most strong correlations are positive.

- Questions 1 and 2 correlate with several other questions, particularly the general competence questions, indicating that perception of attention and competence are linked.

- Similarly, questions 8 and 9 correlate positively to most questions, indicating that musical timing and dynamics are related to enjoyment and attention.





(a) Correlation matrix of averaged scores; green represents a positive correlation, red a negative correlation and black is no correlation

(b) Balloon plot of correlations which are significant at $p < 0.05$; the diameter of circles is proportional to the size of the correlation, and the upper triangle (including main diagonal) cells are left blank. Green circles represent positive correlation, and red are negative

Interactivity Questions

1 ”Did you feel the other participant was paying attention to your playing?”

2 ”Did your partner adopt your ideas in their own playing?”

3 ”Did the other participant surprise you with their playing?”

4 ”Did your partner play their part in a way you had not experienced before?”

5 ”Did your partner introduce new ideas?”

Expressivity Questions

6 ”Did your partner articulate phrases expressively?”

7 ”Did you and your partner stay in time?”

8 ”Did your partner vary their timing in a musical manner?”

9 ”Did your partner use dynamics expressively?”

General Questions

10 ”Did you enjoy playing the excerpt you just played?”

11 ”Would you choose to play with your partner again based on the excerpt you just played?”

12 ”How would you rate your partner’s competence based on the excerpt you just played?”

13 ”Was your partner playing live, or was it a recording of another person?”

(c) Question Texts

Figure 11.5: Visualisations of the correlation matrix for averaged question scores over the entire dataset; in both diagrams, green represents positive correlation and red negative

- Questions 11,12 and 13 have several strong correlations, showing that perception of quality is related to other factors.
- the fact that correlation is not particularly strong inside question groups indicates that the three types of questions are not probing sharply defined, orthogonal factors.

Figure 11.6 shows the average score for each question by condition, and the overall variance in the answers to each question, although nothing of note is concluded from these graphs.

11.4.4 Preliminary Factor Analysis

In order to carry out a successful factor analysis, it is necessary to decide on a number of factors to use ; in this instance, as in many others, there may be a tension between the desired number of factors, and the number suggested by the data. Here, two factors are desired, which could then be mapped onto the concepts expressiveness and interactivity, giving E and I scores.

Breakwell [2006, page 387] suggests two methods of determining the number of factors — examination of eigenvalues, and the *interpretability* of factors with respect to theory — with a strong preference for the latter. However, in the interests of completeness, scree plots of eigenvalues are also presented here[8].

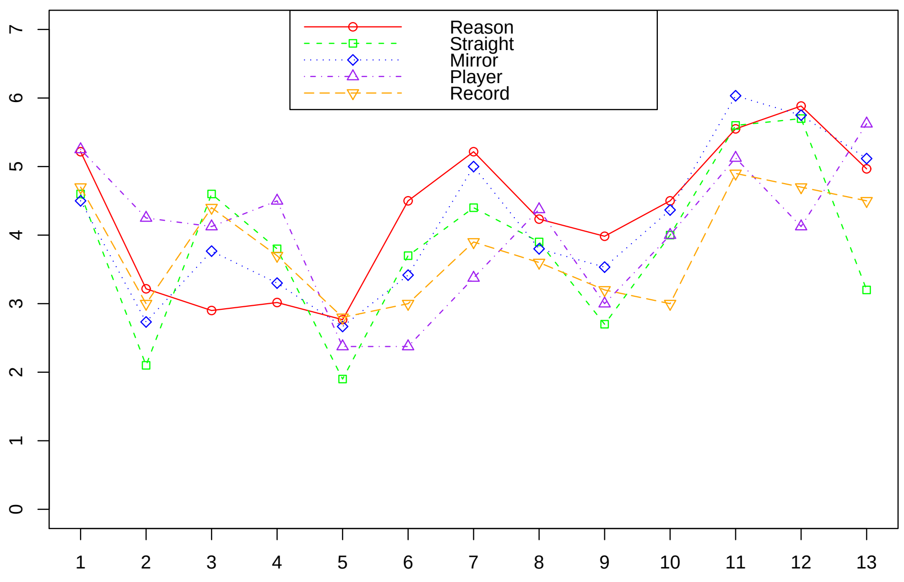
The factor analysis has been carried out in the “R” programming language [R Development Core Team, 2007], using the `factanal` package. This performs maximum likelihood estimation to recover the factor loadings, and also outputs the proportional and cumulative variance for each of the recovered factors. The dataset is only the data from the baseline conditions (Record, Player and Mirror). This ensures that the data with which the model is built is distinct from the data it is later used to explain.

11.4.4.1 Choice of Factors

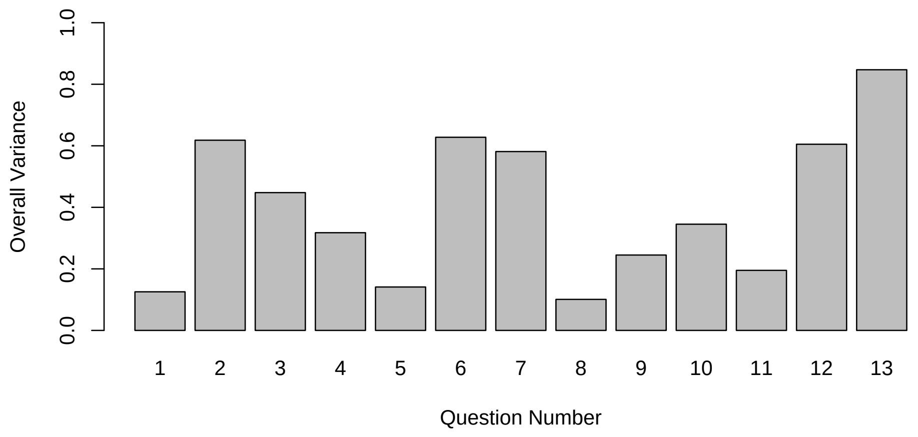
Figure 11.7 shows two different analyses which can inform the choice of the number of factors:

- Figures 11.7(b),11.7(a) show the variance explained by each factor, both individually and cumulatively. It can be noted that there is a relatively large drop-off after the third factor, and that three factors explain more than half of the variance.

[8]a scree plot shows the size of eigenvalues against their index. It gives a visual representation of the importance of factors

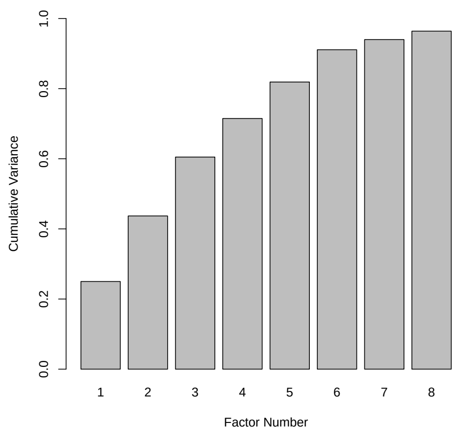


(a) Average score for each question by condition

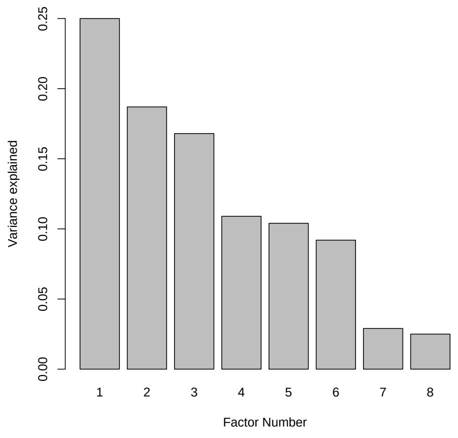


(b) Variance by question

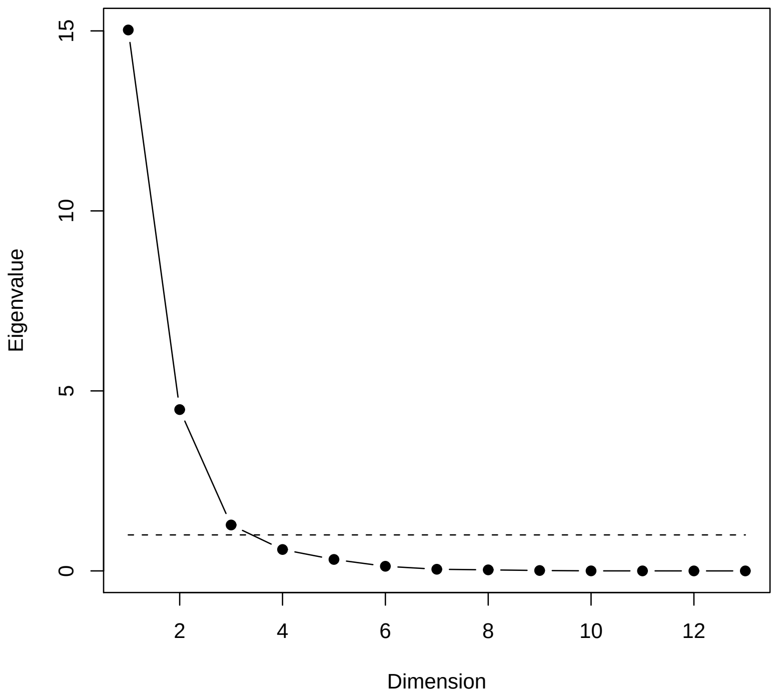
Figure 11.6: Average question scores and question variance for the entire dataset



(a) Cumulative variance for each factor recovered by factor analysis



(b) Variance explained by each factor



(c) Scree plot of the eigenvalues from a PCA decomposition of the data

Figure 11.7: Visualisations of the explanatory power of different factors

- Figure 11.7(c) shows a scree plot of the eigenvalues produced from a Principal Components Analysis (PCA) decomposition of the data [9]. There are two readings of this graph:
  - 3 factors fits the heuristic “choose all eigenvalues greater than 1”.
  - the sharpest change in gradient is between factors 2 and 3

For a more theory oriented approach, the factor loadings when using both 2 and 3 factors are compared, in Figure 11.8. The full loadings are shown, and also the top 4 loadings for each factor, in order to get a feel for which concepts a factor relates to. Finally, Figure 11.9 shows the summed absolute values of the factor loadings for each of the groups of questions.

It now becomes necessary to refer to the intended factor mappings outlined at the start of this chapter, and the hypotheses associated with them. It was hoped that two factors would emerge, one associated with *expressiveness* (E), and one associated with *interactivity* (I). This does not line up with the graph in Figure 11.9, which shows:

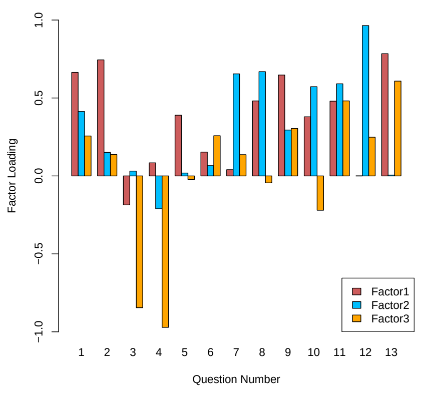
- a first factor, which loads most strongly onto the interactivity questions.
- a second factor, loading onto the general competence and expressivity questions.
- an optional third factor, which loads onto the interactive and general questions.

In light of this, a three factor analysis will be used, since there was not a conclusive argument from looking at variance and eigenvalues, and there is no definite relation to the originally proposed factors. It could be argued that the third factor has a larger difference between the loadings for interactivity and the others, but in response it should be noted that the overall weight of this factor is also much smaller, so it is more desirable to use Factor 1 as I. This assignment of factors should be regarded as tentative at best, and more work would be needed to determine a defensible factor model.

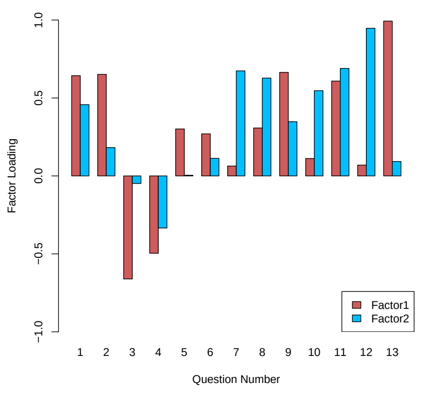
11.4.4.2 Relation of factors to original hypotheses

The results from the Factor Analysis do not support the existence of an “Expressivity” factor; the second factor could more closely be described as “competence”, covering both expressive features, and general “quality of playing” measures. The third factor could be seen as another measure of competence, so the factors will be labelled I, C_1, C_2

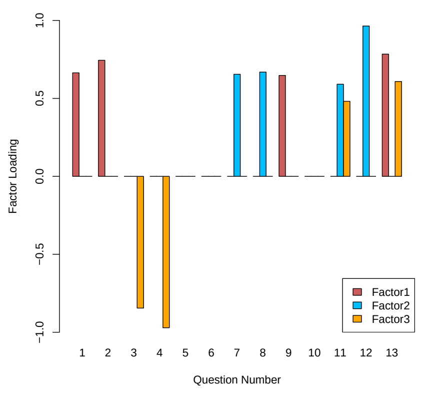
[9] PCA was used since the maximum likelihood estimation module used previously for Factor Analysis does not produce eigenvalues



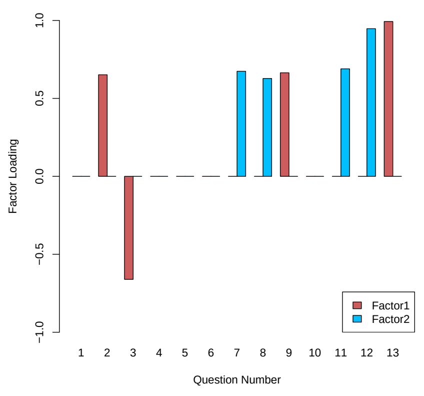
(a) Full factor loadings when 3 factors are used



(b) Full factor loadings when 2 factors are used



(c) Top 4 factor loadings when 3 factors are used



(d) Top 4 factor loadings when 2 factors are used

Interactivity Questions

1 ”Did you feel the other participant was paying attention to your playing?”
2 ”Did your partner adopt your ideas in their own playing?”
3 ”Did the other participant surprise you with their playing?”
4 ”Did your partner play their part in a way you had not experienced before?”
5 ”Did your partner introduce new ideas?”

Expressivity Questions

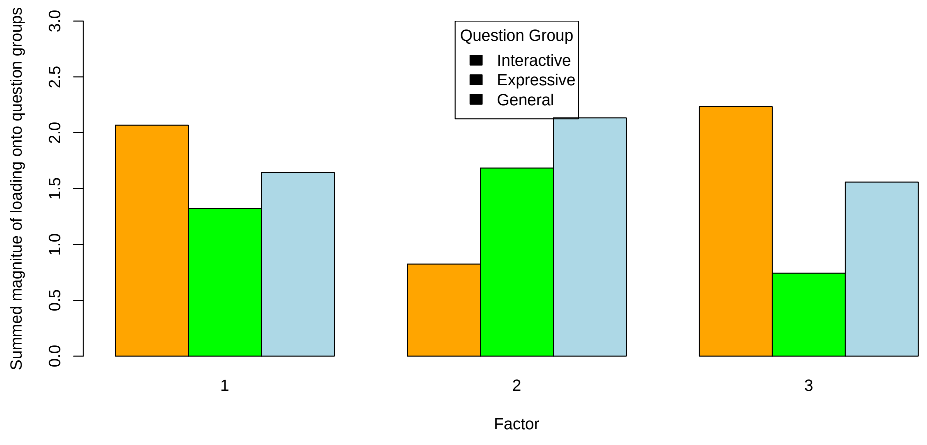
6 ”Did your partner articulate phrases expressively?”
7 ”Did you and your partner stay in time?”
8 ”Did your partner vary their timing in a musical manner?”
9 ”Did your partner use dynamics expressively?”

General Questions

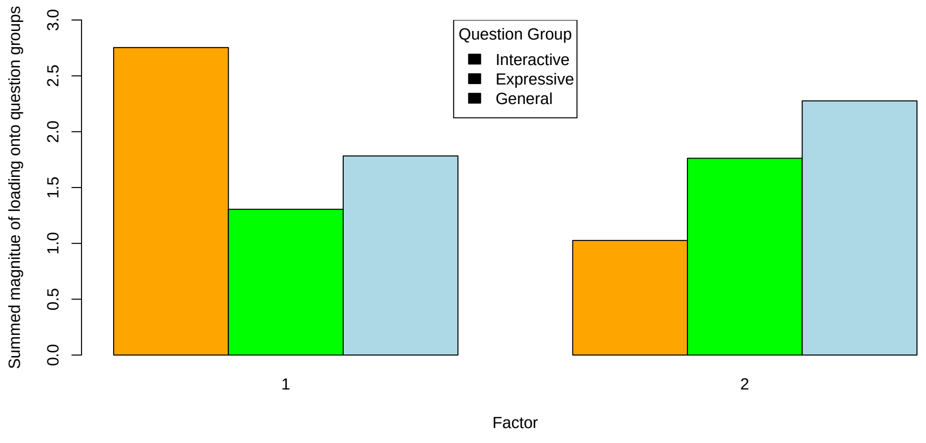
10 ”Did you enjoy playing the excerpt you just played?”
11 ”Would you choose to play with your partner again based on the excerpt you just played?”
12 ”How would you rate your partner’s competence based on the excerpt you just played?”
13 ”Was your partner playing live, or was it a recording of another person?”

(e) Question Texts

Figure 11.8: Factor Loadings by question, for 2 and 3 factors, using Maximum Likelihood estimation, and varimax rotation



(a) Loadings by group for 3 factors



(b) Loadings by group for 2 factors

Figure 11.9: Summed absolute factor loadings by question group

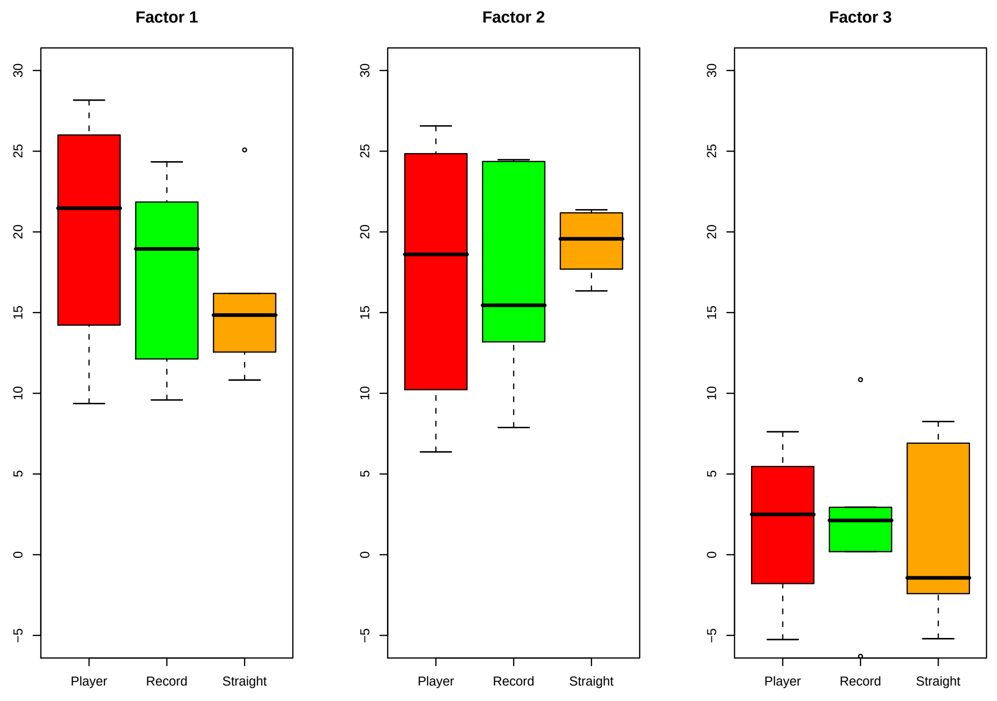


Figure 11.10: Distribution of factor scores for baseline conditions (for 3 factor varimax factor analysis)

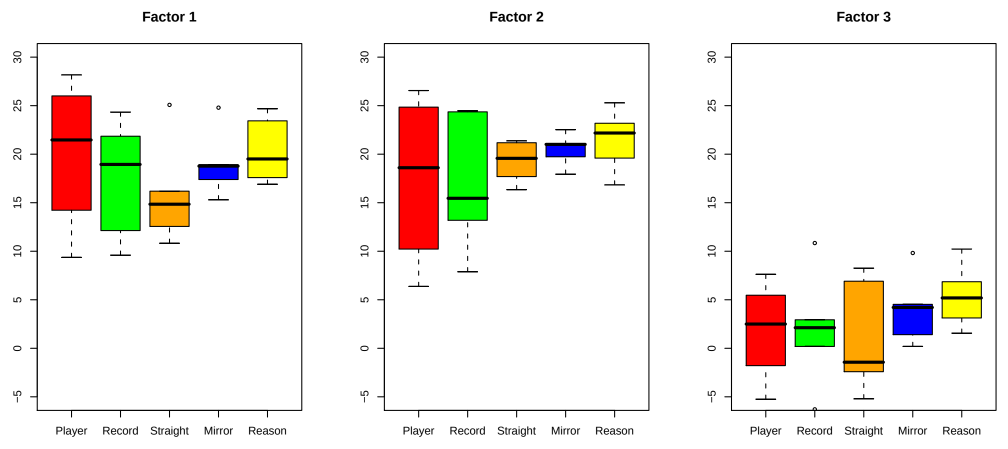


Figure 11.11: Distribution of factor scores for all conditions (for 3 factor varimax factor analysis)

respectively. Since I is the only factor which matches the original hypotheses, this is the only one which will be extensively analysed, although the others will be included on graphs for completeness.

From the initial hypotheses (Sections 11.2.2 and 11.2.3.1), it would be expected that $I_{Player} > I_{Record}$. If it is accepted that the measurement of I is not absolutely perfect, it can also be expected that a recording of a human will be perceived as being slightly interactive, so that $I_{Record} > I_{Straight}$.

In order to test this, it is necessary to reconstruct factor scores for the different conditions. The average answers by person and condition were multiplied by the factor loading matrix to give an average level for each factor, for each person in each condition.

The distribution of these factor levels is shown in Figure 11.10 as box and whisker plots. It can be seen on the graph (although naturally no conclusions may be drawn without further statistical analysis) that $I_{Player} > I_{Record} > I_{Straight}$, as desired (Factor 1). This is not inconsistent with the idea that I is a measure of interactivity.

11.4.5 Comparison of Conditions

In order to investigate the proposed hypotheses, paired t-tests are carried out between the distributions of the factor values produced for each person in each condition, plotted in Figure 11.11.

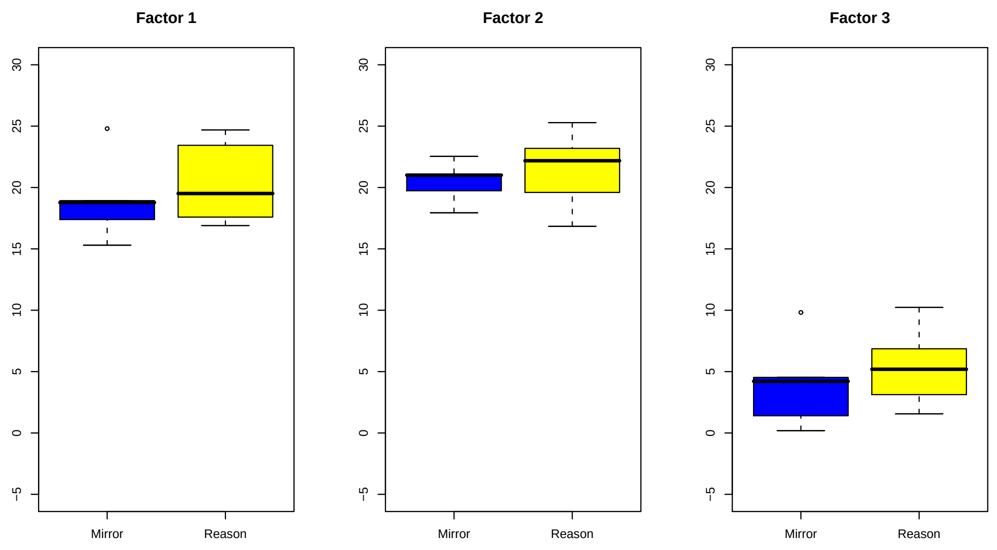


Figure 11.12: Distribution of factor scores for target conditions (for 3 factor varimax factor analysis)

It is now possible to compare the factor scores for the different conditions, to determine the validity of our supporting hypotheses (Section 11.2.2). After removing all of the hypotheses involving E — since no factor was found which looks like a measure of expressivity — the following tests may be performed (using $\alpha = 0.05$).

$I_{Record} < I_{Human}$ performed in unpaired mode, as one of the participants lacks data for I_{Human}, gives t(3)=0.56, p=0.3, which is not significant.

$I_{Straight} < I_{Mirror}$ gives t(4)=1.98, p=0.06, which is almost significant.

$I_{Mirror} < I_{Human}$ gives t(3)=0.38, p=0.36, which is not significant.

It can be expected from the plots of these factor levels that no statistically significant conclusions may be drawn, in large part due to the high variance in the Player and Record conditions, and the t-tests performed back this up. The fact that the Mirror and Straight conditions are almost significantly different is encouraging, as it implies that the participants are sensitive to the musical effects which the system is capable of producing.

There is one overarching question in this whole analysis, which is "does adding the layer of reasoning about musical acts to the system make it feel more interactive?", which has been boiled down to "is $I_{Reason} > I_{Mirror}$?". Figure 11.12 shows the factor scores for the two conditions of most interest. In order to assess whether there is a significant difference, a one-tailed paired t-test with $\alpha = 0.05$ is performed. This gives t(4)=1.26, p = 0.14, which is not significant.

11.4.6 Power Analysis

While it is disappointing that the visible differences are not statistically significant, it is not altogether surprising, given the very small sample size. In order to get a measure of the size of the effect and its relation to the sample size, a brief power analysis may be performed - see Cohen [1988], Cohen et al. [2002] or Cohen [1992] for a concise overview.

The Effect Size is a metric-free measurement of the size of effect, linked to a particular test. For t-tests, "Cohen's *d*" is used [Cohen, 1992], and the convention for values of d is that effect sizes of 0.2 are considered *small*, 0.5 *medium*, and 0.8 *large* [Cohen, 1988, pages 24 – 26].

In this case, looking at the difference in the average scores for Factor 1 for each person between the Reason and Mirror conditions, we have $d = 0.40$ which lies between a medium and small effect.

Using the `power.t.test` in R [R Development Core Team, 2007], there are two calculations to perform:

- given the current dataset, what is the power of the experiment; what is the probability of failing to reject H_0 when an effect of size d is present. Using $n = 5, \alpha = 0.05$, gives $\beta = 0.86$: the experiment has an 86% chance of failing to reject H_0 when there is an underlying effect of the size given — in other words a power of 0.14.

- given the mean and pooled deviation of the dataset, what size of dataset would be necessary to increase the power to a certain level, in this case 0.8 [Cohen, 1988, page56]. Using $\alpha = 0.05, \beta = 0.2$ gives $n = 80$, So 80 participants would be necessary to be able to have a reasonable chance of detecting a true effect, and hence confidence that a lack of significance implies a lack of effect. This number of participants is beyond the resources of this study.

11.5 Discussion and further work

This experiment was performed for two reasons, which should be discussed separately:

- what can be said about the experimental design and implementation in terms of comparing *interactivity* between systems?

- what can be said about the performance of this particular system in the experiment.

11.5.1 Experimental Setup

At the start of the chapter, it was hypothesised that two factors — expressivity and interactivity — would be recovered by the experiment, and that these two factors could then be used to examine the differences between the conditions being tested. Instead, three factors were examined; the first relates to interactivity, and the second two seem to relate to more general notions of competence, or quality of playing. This is not a problem for the overall idea of the experiment, as the presence of the interactive factor is far more important than the expressive factor, but it does raise the question of why the expressive factor was not found. Some possible responses to this are:

- the experimental setup was challenging for some of the participants; as previously noted, issues included not being able to see each other, playing with another person and a metronome, and not having looked at the material in advance. For the experiment to work well, it can be important to have a partner who can play solidly.

- the general questions asked may have been more important to the participants, than the expressive questions, and it would be more appropriate to use a factor based around these ideas of enjoying the interaction and whether the participant would choose to play with that person again.

- the wording of the expressive questions was not clear to all of the participants, and it may be that this caused less accurate answers.

In the future, it would be beneficial to create a standardised battery of questions about interactivity which could be used, although this is too large an undertaking for the current work.

If the setup was challenging to the participants, what could be done to alleviate this? Although the piece was picked so that each participant only had to play a single, repetitive monophonic line, there were still difficulties, mainly centered around having to count many bars accurately. Without having seen a performance of this piece, I would expect that in a standard setting, physical gestures would be extensively used to reduce the need for accurate counting, and extensive rehearsal would allow for a lot of the counting to become very internalised. Since this experiment is only concerned with the interactivity shown, it would have been appropriate to remove the counting altogether, either by giving the participants some signal to indicate changes of section, or by reducing the structure to a single section which could be repeated indefinitely — this was used when collecting performance data to train the deliberative system in Section 9.3.5

Finally, there is the issue that people are being asked to introspect about the interactions they are engaging in; this is a skill which is possessed to varying degrees. Furthermore the answers given depend on the relations between the person, the interaction and the questionnaire, which allows for a large amount of variability. The alternative route would be to develop a computational metric for the quality of interaction, which could then be applied to recordings of each interaction[10]. This would reduce the number of variables involved, and move towards a more objective measurement, but this measurement would then only be as good as the metric which was used. In the future, this is one of the possibilities opened up by Musical Act Theory — see Section 12.2.2 for a discussion.

11.5.2 Performance of the System

The other objective of the experiment was to test the operation of the interaction of the MAMA system, with the possibility of seeing a difference in interactivity caused by the addition of a module based on Musical Act Theory. In terms of the performance of the system, the following can be said:

- All of the experiments proceeded successfully, without any instances of software failure. The system was able to play with people, in realtime, in a reasonably convincing manner.

- Do the features which the system works over capture the expressive components of music to which humans are sensitive? This would be indicated by improved scores for the Reason and Mirror conditions relative to the Straight condition, particularly for the expressive factor. Despite the fact that the second and third

[10]these recordings might be audio, MIDI or some other high level construct

factors could not be directly related to expressiveness, the graphs in Figure 11.11 show the hoped for trend between the three conditions for all factors. Although nothing can be definitively concluded from this, it is an encouraging result.

- Did the addition of the MAT reasoning affect the results? Again, nothing can be concluded definitively, but the difference between the means was in the desired direction.

- Analysing the output of the system after the experiment showed that the reasoner was not performing as hoped; this was both due to an algorithmic bug and a theoretical deficiency. This finding led to the reworking of both the reasoning module (Section 9.4.2) and the theory (Section 6.5.3), so this experiment succeeded in making the system and the theory it is based on better developed and more robust.

- The effect of the bug noted above was that in many cases, the reasoner could not find an appropriate action to carry out — see the discussion in Section 9.4.2, but of the possible opportunities to take some kind of action, only 15% were taken. The resultant behaviour is that when the reasoner does not know what to do, it falls back towards a mirroring behaviour. This means that the performance of the system in this experiment is likely to fall somewhere between an "ideal" Musical Act reasoner and the mirroring system — in other words, the results of the experiment are a lower bound on the utility of Musical Act Theory in this application.

11.6 Conclusion

This chapter has presented a pilot of an experiment, with the following contributions to the thesis:

- an experimental design for assessing the interactivity of similar systems.

- an implementation of this design, and a pilot study using it, which demonstrated some areas which need work, but that the design as a whole is workable.

- feedback about the performance of the system and the theory, which has been used to improve both the system and the theory.

Chapter 12

Conclusions and Future Work

This chapter presents a quick recap of the major claims and achievements presented here, and extends into a discussion of possible future work

12.1 Recap

Architecture The novel term “musical middleware” was coined, to describe a layer of intelligent musical agents which generate musical output based on a score, each other’s actions and input from humans, either musical or extramusical. Several scenarios surrounding the possible uses of this middleware were described, and an architecture for musical agent systems was created. This is described in Chapter 4.

Model of musical interaction A formal system for modelling musical communication was developed in Chapter 6. On a basic level, this models the process of playing and hearing music in a group; the next layer builds on this to model the high level features individual agents can extract from the music they hear, and what they can deduce about the knowledge of the other agents in the system; the final layer builds a set of communicative actions based on this knowledge. This level of Musical Actions provides a foundation on which protocols for interaction can be built, which is independent of the style of music being played, and the particular representation of music used. Possibilities were also sketched for relating these communicative actions to the intentional acts described in Chapter 5.

MAMA In order to test aspects of this theory, an agent system was implemented — MAMA, as detailed in Chapter 7; this system works in real time, and is designed to handle the difficulties of coordinating actions between distributed musicians. MAMA was used in several roles, based on the uses for musical middleware: as a performance tool when playing In C, as a musical companion when playing Canto Ostinato, and as an installation when using the AgentBox as a tactile interface (see Chapter 10 for details). This demonstrated that the basic infrastructure of the system could be used for a variety of tasks, and that it could be used in real time, with human musicians.

Experiment Finally, a central hypothesis was tested (Chapter 11. The hypothesis was:

> H_1: the addition of an understanding of Musical Act Theory to a computational musical agent increases the quality of interaction with human musicians.

A reasoning module was created, which extracts musical actions from the playing of human participants, and then uses sequence completion to find the best action to carry out next, based on a corpus of data collected from human pianists. An experimental setup was created, a cousin of the Turing Test, in which pianists played several short excerpts of a duet, with an unseen partner. Sometimes the partner was a human, or a recording of a human; sometimes a completely straight rendition of the piece; and sometimes the MAMA system, either with or without the Musical Act reasoning module. The quality of each excerpt was rated by the human participant, using a questionnaire decoded with factor analysis. The experiment was run as a pilot study, and highlighted some theoretical issues which were then resolved.

12.2 Evaluation

Section 2.3.3 described a few axes for the classification and evaluation of computer-music systems, which can be used to evaluate different parts of the thesis:

Expressive Completeness/Structural Generality applies to music representation systems. The MRA specification (Section 4.3.7) represents notes at a similar level to MIDI — i.e. pitches, durations etc., so it cannot currently be said to have a high level of expressive completeness; it is designed with extensibility in mind, however, so there is the possibility of adding extra features in. In particular, the current software implementation allows curves in arbitrary features to be implemented, which would allow for a much wider range of expressivity. In terms of structural generality, the representation departs from a linear representation, and allows for scores with a degree of flexibility. Currently, it implements different methods for choosing what musical section to play next, including adding custom decision procedures; this potentially allows for a high level of structural generality, but this has not been fully explored in the current work.

Interactive Systems classifies systems in terms of their drive, response and agency. MAMA as implemented is largely score driven, could be counted as either sequenced or transformative, and is much more of a *player* than an *instrument* system. However, the thrust of the musical agent architecture is very much towards combining score and performance driven systems, and generative/transformative/sequenced modes of output as appropriate at any given time.

Net Music Approaches classify systems as Bridges, Shapers, Servers and Construction Kits, which has been re-interpreted as specifying their temporality and mode of communication. MAMA works on two levels: music is synchronous, with direct communication between software agents, but cannot support direct synchronous musical communication between dislocated humans. Humans can potentially influence the behaviour of remote agents, however, providing synchronous communication through "shared objects"[1]. The underlying architecture is a possible route towards a highly distributed real-time system, and appears to be viable; however, this aspect of the system has not been fully explored and tested.

Interaction Levels define sets of capabilities or activities which constitute a set of graded labels for improvisatory situations. On this scale, the system is currently capable of I_1 (adding extra detail to the score) and I_3 (adding extra measures to the score, in an approved manner). This classification does not capture the complexity of the manner in which extra detail is added, however, so it is only partially relevant. Again, there is the possibility of more involved forms of improvisation — once a set of agents have a set of beliefs about the musical representations being used by their peers, there is a lot of scope for changing and defining the structure of the work dynamically — but this area has not been explored so far.

Creativity classifies systems as being combinatorial, exploratory or transformational; much of MAMA's output is combinatorial, for instance combining the small

[1]although an agent might resent the lack of autonomy implied in being called an object

fragments of music in *In C* in different ways. Some of the output can be said to be exploratory — whenever a new musical value is chosen by the deliberative module, the space of possible values is explored, allowing for novel values from that space. This exploratory mechanism is currently crude, and could be greatly enhanced by the addition of a more informed decision procedure.

Finally, there were several methodological approaches to musical system development and evaluation taken from Pearce et al. [2002]. Due to the scope of this project, these different ideologies are appropriate to different sections of the work, as follows:

Algorithmic Composition is a personally motivated approach, with the evaluation being based on whether the system so created fulfils personal goals. In this case, it is based on my relationship when playing with MAMA, and so will be expanded to included algorithmic performance. I have had a variety of enjoyable interactions with the system, of two main types. Firstly, using the AgentBox allows me to perform In C in a way which is musically interesting to me; I can sculpt the music at a high level, changing texture and instrumentation; I can set up patterns of playing between a group of agents; I understand what effect my actions are likely to have, but there is a complex mapping which means that the system has a sense of autonomy, rather than a simple direct reaction to input. Secondly, when jamming Canto Ostinato, I could play phrases with certain articulation and accents and hear MAMA respond to my playing; again, while it often followed my lead, at other times it would respond with new musical directions — getting quieter as I got louder, replying with alternative patterns of accents etc. So, from a personal perspective, in terms of creating a system which can play with humans, it is a good start. There are many areas which could use expansion, but it fulfils the basic aim of being a non-trivial responsive system.

Theories of musical style is not really relevant to this system.

Compositional Tools relates musical system development to software development methodology, and to my mind relates to the ecological viability of the resultant system. From this point of view, the system has been very successful: an architecture for musical agents was created, which then allowed a formal model to be created and an implementation to be produced. This means that the architectural design has been verified in practice — it has been shown to be implementable, and that it supports a variety of different modes of interaction. The fact that all three components are present has strengthened each one individually.

Cognitive Theories relate computational models to human cognitive processes. It was the aim of the experimental section to provide a cognitive justification for the Musical Actions used in the theory of musical interaction; the results found were encouraging, but no significant conclusions could be drawn from the pilot experiment performed.

12.2.1 Original Aims

The work should also be evaluated with reference to the original set of aims (presented in a different order to their statement in Chapter 1):

> "*produce music, in real time, by interacting and communicating with human musicians.*"

A system was produced, which is capable of playing music in real time, and responding to both musical and extramusical actions. The system was stable enough that it could be used for experiments, and it was also capable of running several simple agents at once on modest computing equipment. Significant work was required to create a system which could implement the various ideas produced as the thesis progressed, and this has resulted in a system with a wide range of capabilities, although some of these capabilities need further expansion in order to be generally applicable — for example, methods for specifying ordering in scores are currently quite "brittle", representations which work on features other than notes can be created, but the agents do not understand them, and the features which the agents use are relatively limited. There is a positive aspect to these limitations, however, which is that they provide a "microworld", which allows a clear perception of the relation between theory and behaviour. In summary, the system meets this goal, although development is always possible.

> "*develop the idea of intelligent musical agents as part of a system of musical middleware, looking at musical avatars, networked music and novel ways of interacting with music.*"

A specification for musical middleware was created, and its evaluation lies in the fact that it was implemented, and supported several different activities. Particular conclusions are:

- The combination of extramusical gestures with communicative musicality is very powerful, as it allows non-musicians access to musical systems, whilst also paving the way for new interaction techniques; the visceral effect of being able to reach out and adjust music which is happening is significant. This has not been experimentally verified, although the reactions of several participants was encouraging, both in terms of interest in playing with the system and in understanding the way in which their actions influenced the music.

- The architecture of the agent system proved to be a suitable platform for combining different modalities — in particular, the addition of a tangible interface to the base agent system was relatively simple, due to the distributed, loosely coupled nature of the agent system. This is extremely encouraging for the possibilities of using this middleware system in a wider range of applications.

Some of the original aims were not thoroughly addressed; in particular, the idea of musical avatars which composers and musicians could create to execute their musical ideas in a networked musical space was not developed here; the system shows some promise in this direction — AgentBox was originally conceived around the idea of musical avatars — but this has not come to fruition. Similarly, the networked aspect of the agent system was assumed — largely through the use of an agent framework which provides network transparent messaging — but not fully explored. So it is a theoretical possibility that the system works in a distributed manner, but it has not been experimentally verified, and issues such as scalability and latency have not been fully worked out.

Overall, the basic aims of musical middleware have been fulfilled, but significant room for development remains.

> "*develop a cognitively plausible model of musical interaction, which is computationally implementable, supports musical agents in reasoning about the actions of others, and is suited to real-time musical applications.*"

A model of musical interaction was developed (Chapter 6), which was found to be computationally implementable; however, the theory was developed more from a formal standpoint than a cognitive one. So, the question of cognitive validity remains. Even if strong evidence had been shown that adding an understanding of this theory improved the performance of an interactive system, this would not be conclusive evidence that the theory relates to cognitive processes in humans. This leads to the question of whether the aim of this work should be to model how humans interact musically, or to create a system which can interact musically with humans; my personal feeling is that the second question is more interesting, and hence that this theory is a step on the road towards an understanding of human music making, even if it is not similar in terms of low level cognitive processes. The emphasis hence shifted towards developing a theory which was internally consistent and as broadly applicable as possible. The fact that this theory has been implemented in a real system gives weight to the idea that it is consistent, implementable and useful. Finally, the use of the theory to sketch a formal semantics for the Musical Acts which were the initial impetus for this work, as well as modelling the musical communication proposed in Pelz-Sherman [1998] is encouraging. Although in both cases sketches were given rather than a fully complete model, it appears that the theory is suitable for these purposes, and provides a technique to describe musical activity in a formal and computational manner which did not previously exist.

> "*develop a method by which the effect of adding an implementation of this theory to a musical system can be quantitatively tested.*"

An experimental design was created to determine precisely this effect, in a quantitative manner. The experiment was implemented as a pilot study, due to a number of unknowns and the difficulty of procuring pianists. The experiment assumed the existence of two factors, interactivity and expressivity; no strong evidence for these was found, and the two factors recovered were closer to interactivity and competence, but this could not be strongly claimed. The effect of the addition of the theory also could not be seen. So as a pilot study, the conclusions which could be made were that a fuller investigation of the experimental design is needed — in particular a model which includes competence as a factor — along with more work on the battery of questions used. Progress has been made towards this aim, but it cannot be said to have been fulfilled.

> "*Explore the social aspects of music, and develop a clearer understanding of the communicative properties of music making.*"

The work carried out here has been inspired by the conception of music as a communicative process; the question is then how much this conception has been explored and implemented. The theory developed explicitly models musical communication, but little attention has been paid to the more social aspects of this — for example, the implications of this musical communication between groups of agents, both within interactions and evolving across multiple interactions, or the relations between humans and musical agents, particularly when engaged in networked music creation. Overall, the ideas presented here provide a starting point from which the social aspects of musical communication can be explored, but this work does not explore the ramifications of these ideas.

12.2.2 Overall conclusion

Looking at the conclusions for each of the aims listed does not paint a highly positive picture of the work, as most of the aims were only partially met. This must be taken in the context of the range and ambition of the original aims, and the multidisciplinary nature of the work. Since the plan was to create several related components, it is inevitable that each component is less complete than desired. The strength then comes from the combination of all these different approaches: the middleware architecture which gives rise to a theoretical model, which is implemented by a computational system. The partial satisfaction of the aims is a consequence of the scope and novelty of this work, which provides a broad foundation for increasing the computational understanding of human musical communication and encouraging the use of this understanding in real world projects.

12.3 Future Work

This thesis has provided a sketch of techniques which could potentially be applied in many different ways, but each of which requires significant further work.

Communication in Human Music Making: The original thrust of this theory was to be a computational implementation of the kind of musical communication carried out by humans; this proved to be a slightly ambitious goal, as there did not exist a clear, structured approach to the analysis of improvised music which is suitable for computational analysis. Hence, the theory developed as much from a logical standpoint as a cognitive one. Since the ultimate test of any theories about music making has to be the relation of the theory to the practice of making music, it would be of great interest to apply this theory to a wide range of musical situations, and examine how the analysis aids understanding of the dynamics involved. As the corpus is built up, patterns may be extracted: particular sequences of musical actions which are commonly used. There is then the work of comparing the usage of different patterns:

- by different people,

- by different roles in the performance (e.g. soloist, rhythm section),
- to the intentions of the performers,
- across different styles and cultures of music making.

As well as helping understand the process of human improvisation, this would enable virtual improvisers to create musically sensitive, stylistically appropriate playing. Agents would be able to take on particular roles at different times in the playing, and tailor their response making to the task at hand. Also, a model relating musical actions to intention could be built up, to allow for the musical surface to be interpreted in terms of performative Musical Acts.

Interactivity Measure: Building on the the idea of analysing a corpus of human improvisations, there is the possibility of creating a computationally implementable measure of interactivity, which can be applied to performances, both during creation and post-hoc, as follows:

- a corpus of musical performances are created, and questionnaires at the end of each excerpt are used to get a score for their interactivity.
- the sequences of Musical Actions which gave rise to these scores are extracted, allowing sequences of Musical Actions to be given a partial score, or particular features of the act stream to be used as predictors of the interactivity score.
- new musical performances can then be run through an act extractor, and have a score generated for their interactivity.

Such a system would have a wide variety of applications; it could provide a metric for comparing musical systems; it could provide a score or fitness function for machine learning systems to use when making music; it could be a new way to index general musical content.

A system which could produce realtime musical act analysis of group performance would have an even wider range of applications. It would allow teachers to understand their lessons with students — particular patterns of Musical Actions could then be moved towards or avoided, and the teacher could see a relation between teaching methods and transfer of musical ideas. Bands could analyse the way they play together, and work towards a more interactive, collaborative style of playing.

Interactive Musical Interfaces: The current tangible interface to the system, while exciting, is more of a proof of concept than a final product. There is a rich area of possibility opened up by having a set of intelligent agents which react to physical input. The current system of communication could be expanded, with reference to human extramusical gestures, to allow a fuller range of gestures and gesture types. As well as the relatively "formal" gestures currently used, a variety of more "expressive" gestures could be captured from human body movement and injected into the agent system; the infrastructure would make this relatively easy: the challenge is in choosing what gestures to interpret, and how the agents might react to these.

Distributed Jamming: One of the most complex of the paradigms suggested for Musical Middleware is that of Distributed Jamming, where several agents work together to alter a shared score, with or without human interference. This would build on the ideas introduced in the development of Musical Act Theory about Common Acceptance, but would require on top of this a layer of reasoning which could create ad hoc musical structures and make decisions about:

- which section of the structure is being played at the moment
- when a rendition of a section is sufficiently different from the current value to become a new section in its own right
- choosing an appropriate sequence of sections

and so on; this would provide a powerful compositional tool - a player could work with a band of (virtual) accomplices, who would adapt the ideas presented to their own particular styles of playing, and help create (or at least elucidate) structures.

Responsive Accompaniment: At present, the MAMA system is not able to generate notes, and must rely on a detailed score. There are existing systems, which take high level representations, and generate accompaniment; the most famous of these is "Band in a Box"(BiB) [Gannon, 1991], which produces MIDI accompaniments. However, BiB does not have a lot of real time functionality, and is not very responsive to play with. The possibility then arises of using the BiB system to generate notes, while MAMA is used to manage the interactive aspects of the performance. Some different levels of operation would be:

- BiB generates notes according to the predefined structure; MAMA's existing expressive analysis and generation is then used to modify the output in response to the output of human performers. This would require very little change to MAMA - simply replacing reading from a score with input from BiB.
- MAMA could be allowed to interfere in BiB's note generation process; for example, when BiB is choosing drum rhythms or keyboard patterns to use, MAMA could use the accent structures extracted from human playing to influence this choice and pick patterns which work with the current performance.
- MAMA's harmonic capabilities could be extended to analysing the chord sequences which humans were playing, and allow the creation of musical structure "in the moment", so that BiB was used to provide notes for particular chords, but MAMA managed the musical and harmonic structure in response to the rest of the musicians.

12.4 Contributions

The major contributions which have been presented in this thesis are:

- creation of a computational model of musical interaction, which includes a system of musical actions which describes the communicative content of the music.
- design of an architecture based on this model, and implementation of a system which embodies this architecture.
- three case studies demonstrating the use of the system for different tasks.
- an experimental design and pilot study examining the hypothesis that adding an understanding of the model of musical interaction to a system will improve its interactivity when playing with humans.

Appendix A

Map of MAMA source code

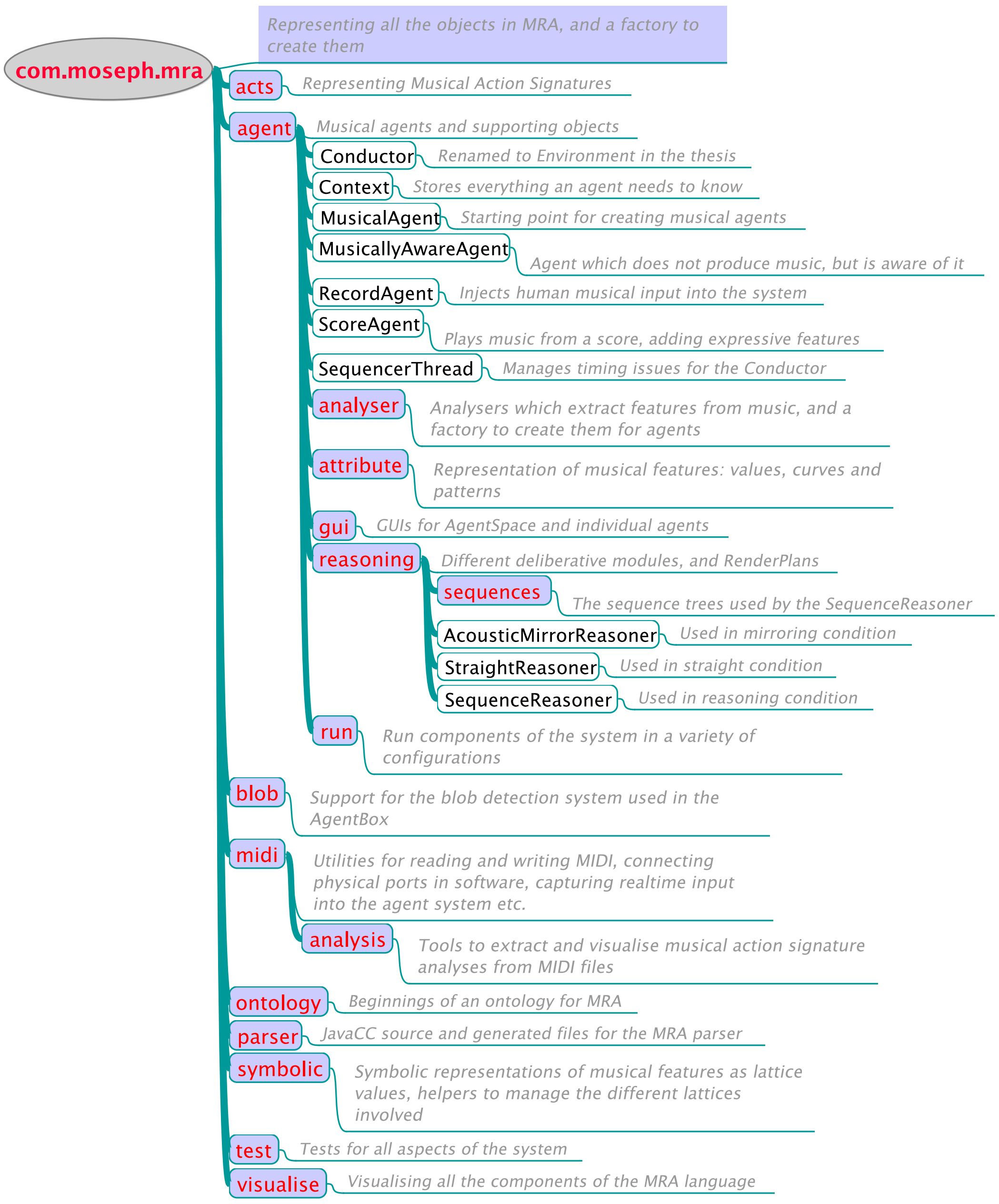

com.moseph.mra
Representing all the objects in MRA, and a factory to create them
acts
Representing Musical Action Signatures
agent
Musical agents and supporting objects
Conductor
Renamed to Environment in the thesis
Context
Stores everything an agent needs to know
MusicalAgent
Starting point for creating musical agents
MusicallyAwareAgent
Agent which does not produce music, but is aware of it
RecordAgent
Injects human musical input into the system
ScoreAgent
Plays music from a score, adding expressive features
SequencerThread
Manages timing issues for the Conductor
analyser
Analysers which extract features from music, and a factory to create them for agents
attribute
Representation of musical features: values, curves and patterns
gui
GUIs for AgentSpace and individual agents
reasoning
Different deliberative modules, and RenderPlans
sequences
The sequence trees used by the SequenceReasoner
AcousticMirrorReasoner
Used in mirroring condition
StraightReasoner
Used in straight condition
SequenceReasoner
Used in reasoning condition
run
Run components of the system in a variety of configurations
blob
Support for the blob detection system used in the AgentBox
midi
Utilities for reading and writing MIDI, connecting physical ports in software, capturing realtime input into the agent system etc.
analysis
Tools to extract and visualise musical action signature analyses from MIDI files
ontology
Beginnings of an ontology for MRA
parser
JavaCC source and generated files for the MRA parser
symbolic
Symbolic representations of musical features as lattice values, helpers to manage the different lattices involved
test
Tests for all aspects of the system
visualise
Visualising all the components of the MRA language

Figure A.1: Map of MAMA source code

Appendix B

Derivation of two agent value relations

Figure B.1 shows the relations of interest between the three values involved when agent *A* produces a new value for a single facet of the musical surface.

If $R = Same, Subsumes, Subsumed, Alter, Disjoint$ is the set of possible relations, it could be expected that since each relation can take one of these five values, the possible set of three way relations is the Cartesian cube of this set, or R^3, with 125 members. However, due to structural considerations on the relations between the values, and the context in which we are interested in these values, this may be reduced significantly.

Firstly, it should be noted that `SAME` is not an allowable value for R_{self}, as this would not count as the execution of an action under the present formulation.

This derivation will proceed as follows:

- select a value for R_{prev}.
- given R_{prev}, iterate over possible values of R_{self}, deciding which values of R_{other}

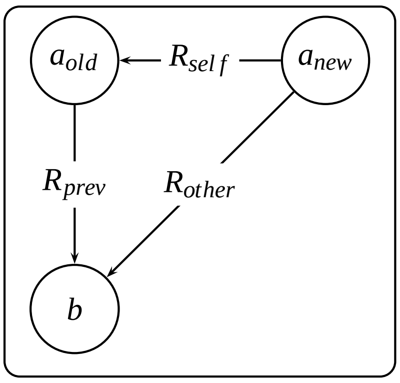


Figure B.1: Relations between the values of two agents *a* and *b*, from the point of view of *a* constructing a new value

$R_{prev} = \texttt{SAME}$

R_{self}	Allowed values for R_{other}
SUBSUMED	SUBSUMED
SUBSUMES	SUBSUMES
ALTER	ALTER
DISJOINT	DISJOINT

$R_{prev} = \texttt{SUBSUMED}$

R_{self}	Allowed values for R_{other}
SUBSUMED	SUBSUMED
SUBSUMES	SAME,SUBSUMED,SUBSUMES,ALTER,DISJOINT
ALTER	SUBSUMED,ALTER,DISJOINT
DISJOINT	DISJOINT

$R_{prev} = \texttt{SUBSUMES}$

R_{self}	Allowed values for R_{other}
SUBSUMED	SAME,SUBSUMED,SUBSUMES,ALTER
SUBSUMES	SUBSUMES
ALTER	SUBSUMES,ALTER
DISJOINT	SUBSUMES,ALTER,DISJOINT

$R_{prev} = \texttt{ALTER}$

R_{self}	Allowed values for R_{other}
SUBSUMED	SUBSUMED,ALTER
SUBSUMES	SUBSUMES,ALTER,DISJOINT
ALTER	SAME,SUBSUMED,SUBSUMES,ALTER,DISJOINT
DISJOINT	SUBSUMES,ALTER,DISJOINT

$R_{prev} = \texttt{DISJOINT}$

R_{self}	Allowed values for R_{other}
SUBSUMED	SUBSUMED,ALTER,DISJOINT
SUBSUMES	DISJOINT
ALTER	SUBSUMED,ALTER,DISJOINT
DISJOINT	SAME,SUBSUMED,SUBSUMES,ALTER,DISJOINT

Figure B.2: Possible Relationships produced by Prolog program

are possible

This is performed in two different manners: using a Prolog program to generate the results, and by hand, explained with natural language reasoning.

B.1 Automatic Generation

Since it is highly possible to make mistakes in this type of analysis, a more rigorous approach is appreciated. To this end, a Prolog program was created (with a perl script to manage it) which generates the possible relations. In order to deal with Prolog's negation-as-failure, a set of facts can be derived from each relation, which can then be checked for incompatibilities. The output of the program is summarised in Figure B.2.

B.2 Manual Derivation

B.2.1 $R_{prev} =$ **`SAME`**

If a's current value is the same as b's current value, then necessarily $R_{self} = R_{other}$. Hence, the following possibilities are allowed:

$R_{prev} =$ `SAME`

R_{self}	Allowed values for R_{other}
`SUBSUMED`	`SUBSUMED`
`SUBSUMES`	`SUBSUMES`
`ALTER`	`ALTER`
`DISJOINT`	`DISJOINT`

B.2.2 $R_{prev} =$ **`SUBSUMED`**

If a_{old} is subsumed by b, then in graph terms, we know that a_{old} has b as an ancestor. This guides the following deductions for different values of R_{self}:

`SUBSUMED` means that a_{old} is an ancestor of a_{new}, hence b must also be an ancestor of a_{new}, hence the only possible value for R_{other} is `SUBSUMED`

`SUBSUMES` makes a_{old} a descendant of a_{new}; this allows for construction of situations in which all possible values for R_{other} hold.

`ALTER` means that a_{new} cannot be an ancestor of a_{old}, ruling out `SAME` and `SUBSUMES`.

DISJOINT requires that there is no common ancestor between a_{new} and a_{old}, which must also mean that a_{new} is not an ancestor of b, so SUBSUMES is ruled out, as this would make a_{new} and ancestor of b, and hence of a_{old}. ALTER is ruled out as this would make a common ancestor between b and a_{new}, and by extension between a_{old} and a_{new}. SUBSUMED would make b a common ancestor between a_{old} and a_{new}. SAME is clearly not a possibility.

$R_{prev} =$ SUBSUMED

R_{self}	Allowed values for R_{other}
SUBSUMED	SUBSUMED
SUBSUMES	SAME,SUBSUMED,SUBSUMES,ALTER,DISJOINT
ALTER	SAME,SUBSUMED,SUBSUMES,ALTER,DISJOINT
DISJOINT	DISJOINT

B.2.3 $R_{prev} =$ SUBSUMES

If a_{old} subsumes b, then a_{old} must be an ancestor of b, so the following can be deduced for different values of R_{self}:

SUBSUMED means a_{old} is an ancestor of a_{new}, so it will always be a common ancestor for b and a_{new}, ruling out DISJOINT.

SUBSUMES means a_{new} is an ancestor of a_{old}, and hence of b, so only SUBSUMES is allowed.

ALTER rules out SUBSUMED as this would make a_{new} a descendant of a_{old} (via b). Also rules out SAME

DISJOINT means that a_{new} cannot have a common path with a_{old}, hence it cannot be a descendant of a_{old} (rules out SAME,SUBSUMED), but it may be an ancestor, or share an ancestor.

$R_{prev} =$ SUBSUMES

R_{self}	Allowed values for R_{other}
SUBSUMED	SAME,SUBSUMED,SUBSUMES,ALTER
SUBSUMES	SUBSUMES
ALTER	SUBSUMES,ALTER
DISJOINT	SUBSUMES,ALTER,DISJOINT

B.2.4 $R_{prev} =$ **ALTER**

If a_{old} `ALTER` s b, then they must share a common ancestor, but may not be ancestors or descendants of each other. The deductions for different values of R_{self} are:

SUBSUMED means that a_{new} must be a descendant of a_{old}; it hence cannot *SUBSUME* b (as that would make b a descendant of a_{old}), and it cannot be `DISJOINT`, as there is already a common ancestor between a_{old} and b, and a_{old} is an ancestor of a_{new}; a_{new} cannot be the `SAME` as b, as b cannot be a descendant of a_{old}.

SUBSUMES makes a_{old} a descendant of a_{new}; a_{new} cannot be subsumed by b, as that would make a_{old} an ancestor of b, and it cannot be the `SAME` as b, as that would make b subsume a_{old}.

ALTER allows all possibilities

DISJOINT means that there is no common ancestor between a_{old} and a_{new}; this means that a_{new} cannot be a descendant of b (or the `SAME`) as this would provide a common ancestry.

$R_{prev} =$ `ALTER`

R_{self}	Allowed values for R_{other}
SUBSUMED	SUBSUMED,ALTER
SUBSUMES	SUBSUMES,ALTER,DISJOINT
ALTER	SAME,SUBSUMED,SUBSUMES,ALTER,DISJOINT
DISJOINT	SUBSUMES,ALTER,DISJOINT

B.2.5 $R_{prev} =$ **DISJOINT**

Here, there can be no common ancestry between a_{old} and b. Looking at different values for R_{self}, this means that:

SUBSUMED makes a_{new} a descendant of a_{old}; it may have b as an ancestor, but may not be a descendant of (or the `SAME`) as b.

SUBSUMES makes a_{new} an ancestor of a_{old}; since there is no common ancestry between a_{old} and b, a_{new} cannot have any common ancestry with b either (as this would provide a point of common ancestry between a_{old} and b).

ALTER means a_{old} and a_{new} share a common ancestor. a_{new} cannot be an ancestor of b (violates $R_{prev} =$ `DISJOINT`).

DISJOINT places no restrictions on the relation between b and a_{new}, as they are both DISJOINT from a_{old}.

$R_{prev} =$ DISJOINT

R_{self}	Allowed values for R_{other}
SUBSUMED	SUBSUMED,ALTER,DISJOINT
SUBSUMES	DISJOINT
ALTER	SUBSUMED,ALTER,DISJOINT
DISJOINT	SAME,SUBSUMED,SUBSUMES,ALTER,DISJOINT

B.3 Descriptions of Musical Actions

In order to aid understanding of what a musical action signature means, Figure B.1 presents a table of all possible action signatures, along with a simple diagram which illustrates a possible graph-fragment corresponding to that signature. The graph uses the nodes {b,o,n} to stand for the other agent's value (b stands for Agent B), o for the old value and n for the new value. The nodes x,y,z are used when a node needs to exist which is not $\odot$, and not otherwise defined — mostly to provide a common ancestor for ALTERS relations. Finally, the node $\odot$ has its usual meaning as the top of the lattice structure, except in two cases where it appears twice in the graph; this is purely for typographical reasons, and the diagram should be read as if both of the $\odot$ nodes were the same node — connections from the lower copy of $\odot$ should be assumed to come from the top version.

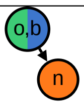
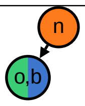
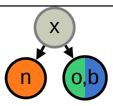
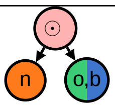
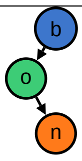
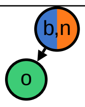
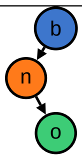
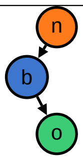
R_{self}	R_{other}	R_{pref}	Diagram	Description
subsumed	subsumed	same		A more elaborated or specified version of the current value
subsumes	subsumes	same		A simplification of the current value
alter	alter	same		Moving from a shared value to a new, but related direction
disjoint	disjoint	same		Moving from a shared value to a totally new idea
subsumed	subsumed	subsumed		Moving from an elaboration to a further elaboration; extending further
subsumes	same	subsumed		Pulling back from an extension to share the same value
subsumes	subsumed	subsumed		Less elaborate, but still more elaborate than B
subsumes	subsumes	subsumed		Simpler than either previous value

Table B.1: Complete table of possible action signatures with descriptions and illustrations

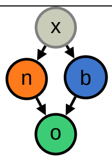
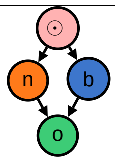
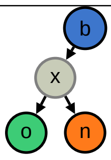
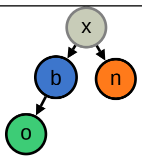
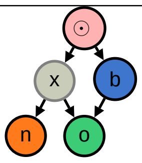
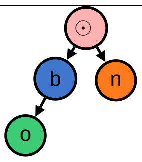
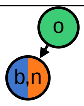
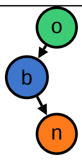
R_{self}	R_{other}	R_{pref}	Diagram	Description
subsumes	alter	subsumed		Removing some of the commonality; simplifying back towards a different direction
subsumes	disjoint	subsumed		Simplifying towards a very different direction
alter	subsumed	subsumed		Trying a different extension
alter	alter	subsumed		An alternative direction, based on a simplification of B
alter	disjoint	subsumed		Moving away from B
disjoint	disjoint	subsumed		Moving from an elaboration of B to a totally new direction
subsumed	same	subsumes		Joining B in an elaboration
subsumed	subsumed	subsumes		Developing B's elaboration further

Table B.1: Complete table of possible action signatures with descriptions and illustrations

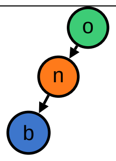
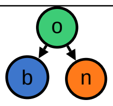
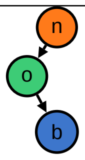
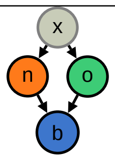
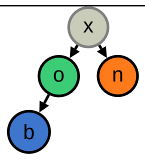
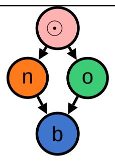
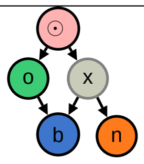
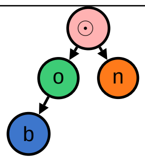
R_{self}	R_{other}	R_{pref}	Diagram	Description
subsumed	subsumes	subsumes		Moving towards B's elaboration
subsumed	alter	subsumes		An alternative to B's elaboration
subsumes	subsumes	subsumes		Simplifying further
alter	subsumes	subsumes		Finding an alternative simplification of B
alter	alter	subsumes		A different elaboration to either previous value
disjoint	subsumes	subsumes		A very different simplification of B
disjoint	alter	subsumes		An elaboration of an alternative simplification of B
disjoint	disjoint	subsumes		New musical direction

Table B.1: Complete table of possible action signatures with descriptions and illustrations

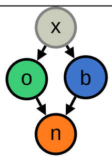
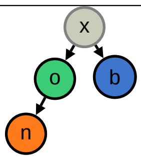
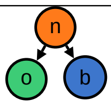
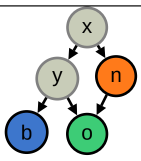
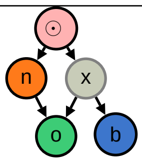
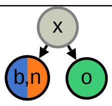
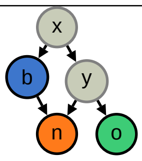
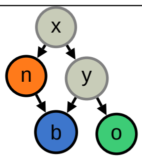
R_{self}	R_{other}	R_{pref}	Diagram	Description
subsumed	subsumed	alter		A synthesis of previous values
subsumed	alter	alter		A further elaboration, ignoring B
subsumes	subsumes	alter		A simplifying synthesis
subsumes	alter	alter		A simplification with some relation to B
subsumes	disjoint	alter		A simplification with no relation to B
alter	same	alter		Joining B on a related idea
alter	subsumed	alter		A related value which is an elaboration of B
alter	subsumes	alter		A related value which is a simplification of B

Table B.1: Complete table of possible action signatures with descriptions and illustrations

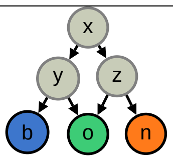
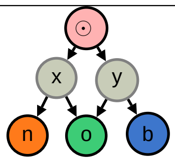
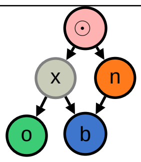
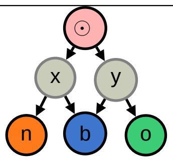
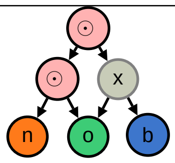
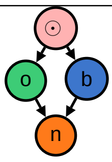
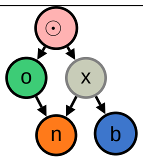
R_{self}	R_{other}	R_{pref}	Diagram	Description
alter	alter	alter		A new elaboration of the commonality between previous values
alter	disjoint	alter		A new elaboration which is totally different to B
disjoint	subsumes	alter		A simplification of B, with no relation to the previous value
disjoint	alter	alter		An alternative version of B, with no relation to previous value
disjoint	disjoint	alter		A totally new direction, where previously there was commonality
subsumed	subsumed	disjoint		A synthesis of two previously unrelated values
subsumed	alter	disjoint		An elaboration which creates a commonality with B

Table B.1: Complete table of possible action signatures with descriptions and illustrations

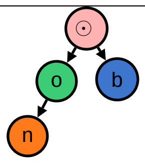
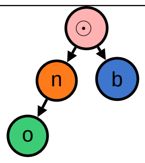
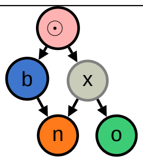
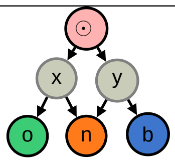
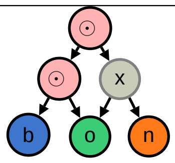
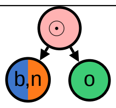
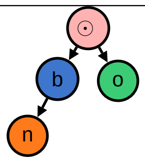
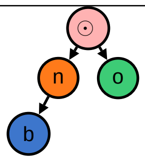
R_{self}	R_{other}	R_{pref}	Diagram	Description
subsumed	disjoint	disjoint		Elaboration of a disjoint value
subsumes	disjoint	disjoint		Simplification of a disjoint value
alter	subsumed	disjoint		Elaboration of B which has commonality with previous value
alter	alter	disjoint		New value which is related to both previously unrelated values
alter	disjoint	disjoint		Related to previous value, but still unrelated to B
disjoint	same	disjoint		Joining B on an unrelated value
disjoint	subsumed	disjoint		Elaboration of B's unrelated value
disjoint	subsumes	disjoint		Simplification of B's unrelated value

Table B.1: Complete table of possible action signatures with descriptions and illustrations

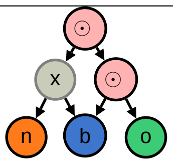
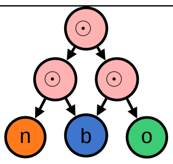
R_{self}	R_{other}	R_{pref}	Diagram	Description
disjoint	alter	disjoint		Related value to B's unrelated value
disjoint	disjoint	disjoint		Unrelated value from unrelated values

Table B.1: Complete table of possible action signatures with descriptions and illustrations

B.4 Example Musical Act Analysis

Time (s)	Instrument	Performative	Description
0:00 - 0:09	Bass, Guitar	PROPOSE	Bass + Electric Guitar establish a tonal centre. Bass suggests tonic, and guitar replies with n extenison.
	Bass, Guitar	CONFIRM	Repeated as a confirmation, and then the guitar suggests an alternative extension
0:09 - 0:34	Cuica	CONFIRM	Cuica confirms the current beat
	Tablas	PROPOSE	Tablas suggest a “1 2+” rhythm
0:34 - 0:55	Trumpet	PROPOSE	Trumpet suggests a scale 1 and a lyrical phrase
	Clarinet	CONFIRM	Clarinet echoes the same scale, in a similarly lyrical style
	Xylophone	EXTEND	Xylophone follows the given scale, but adds dissonant elements
	Backing	None	Backing abandons complex rhythm
0:55 - 1:08	Trumpet	EXTEND	Exploring scale 1
	Clarinet	CONFIRM	Follows trumpet’s exploration of scale 1
	Triangle	PROPOSE	Suggests a more explicit rhythm
	Trumpet	CONFIRM	Spiky notes agree with triangle’s suggestion
1:08 - 1:35	Triangle	REQUEST	A loud clang signals the start of a new section
	Clarinet	PROPOSE[1]	A new scale 2 still in a lyrical style
	Trumpet	REJECT	By sticking with scale 1, the trumpet rejects the clarinet’s proposal
1:35-1:47	Xylophone	CONFIRM	Xylophone confirms clarinet’s scale 2

Table B.2: Musical Act Analysis of Little Blue Frog, by Miles Davis

[1]Repeated suggestion, so maybe followed by PROPOSE-AGAIN?

Time (s)	Instrument	Performative	Description
	Trumpet	CONFIRM[2]	Trumpet gives in to the new scale, and leaves with a few parting blasts
1:47 - 2:00	Triangle	EXTEND	Many muted notes extend and emphasise the rhythmic ideas
	Trumpet	EXTEND	Using scale 2, the trumpet adds an increasing rhythmic element
	Clarinet	PROPOSE	Clarinet introduces a new scale 3 (More eastern sounding), ignores the trumpet's rhymic direction and continues lyrically
2:00 - 2:13	Trumpet	PROPOSE	A spiky, stabbing phrase, based on scale 2
	Clarinet	CONFIRM	briefly seems to agree with the trumpet.
2:13 - 2:29	Bass clarinet	CONFIRM	Confirms scale 3
	Trumpet	REJECT	Ignores bass clarinet, and continues with stabs
	Clarinet	REJECT	Ignores bass clarinet, and continues with lyricism in scale 2
2:29 - 2:43	Trumpet Clarinet Bass clarinet	ARGUE	All play lyrically, with clarinet on scale 2, trumpet on 1 and Bass Clarinet in 3
2:43 - 3:08	Trumpet	PROPOSE	proposes a resolution, by playing stabs which fit with any of the scales
3:03 - 3:08	E-Piano Vibes	CONFIRM	supports the trumpet's resolution
3:08 - 3:17	Backing	REJECT	Increased dissonance and rhythmic confusion reject the proposed resolution

Table B.2: Musical Act Analysis of Little Blue Frog, by Miles Davis

[2]Also withdraws - do we need a WITHDRAW?

Time (s)	Instrument	Performative	Description
3:17 - 3:35	All	CONFUSION	
3:55 - 4:11	Snare	REQUEST	Snare enters to call for new section with crescendoing 8th notes
	Winds	CONFIRM	Winds join in with the 8th note idea
4:08 - 4:11	Triangle	CONFIRM	A bar of loud crotchets pinpoints the section change called for by the snare drum

Table B.2: Musical Act Analysis of Little Blue Frog, by Miles Davis

The performatives used in this analysis were:

PROPOSE
CONFIRM
REJECT
EXTEND
ALTER
REQUEST
ARGUE

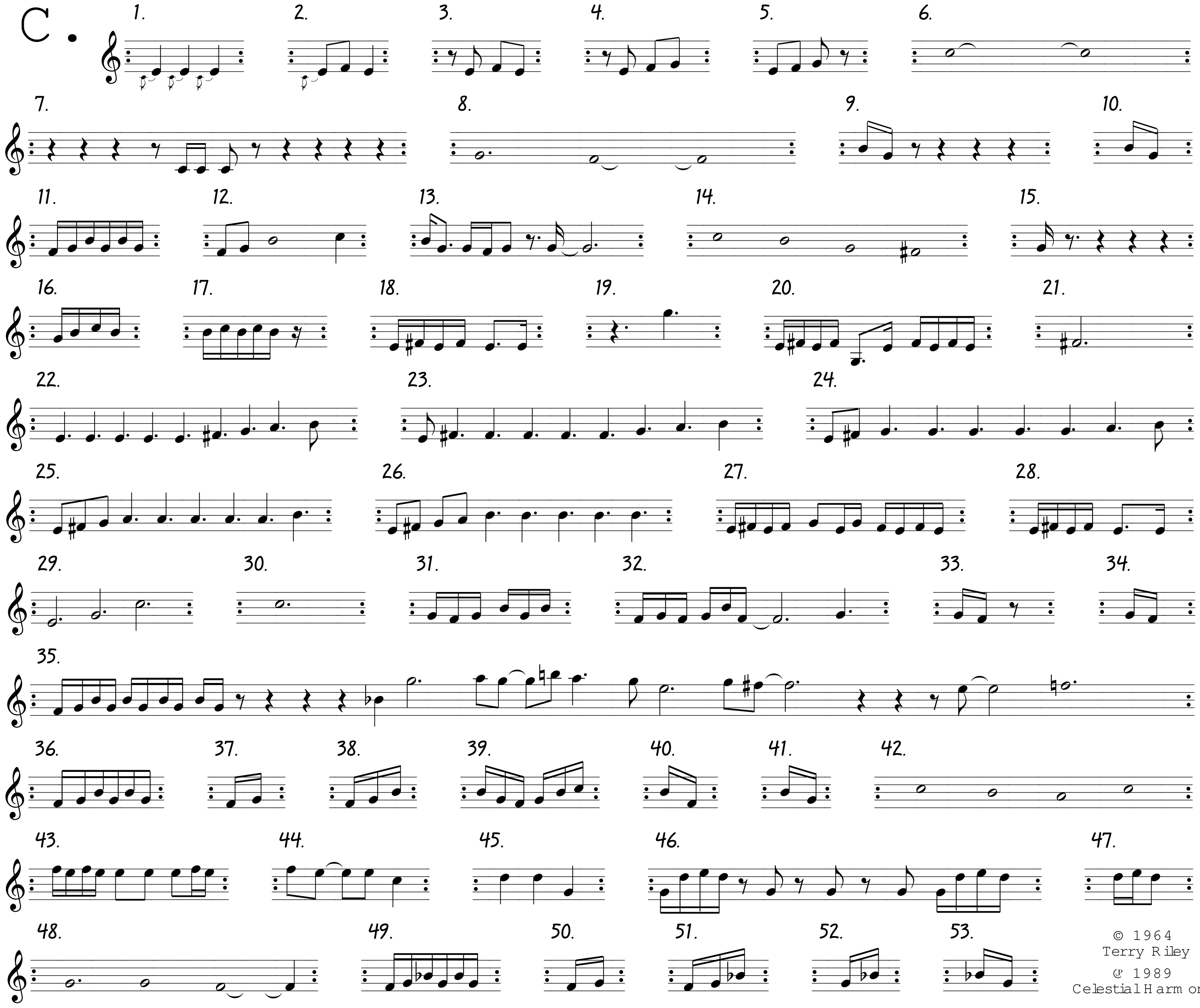
Appendix C

In C

C.1 Complete Score

The following pages show the complete score for In C, which is currently available from: http://www.otherminds.org/SCORES/InC.pdf. The score is ©1964 Terry Riley; permission has been asked from the publishers to reproduce this here. This section does not fall under the Creative Commons licence of the rest of the thesis.

in C.


1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.
12.
13.
14.
15.
16.
17.
18.
19.
20.
21.
22.
23.
24.
25.
26.
27.
28.
29.
30.
31.
32.
33.
34.
35.
36.
37.
38.
39.
40.
41.
42.
43.
44.
45.
46.
47.
48.
49.
50.
51.
52.
53.

In C

Performing Directions

All performers play from the same page of 53 melodic patterns played in sequence.

Any number of any kind of instruments can play. A group of about 35 is desired if possible but smaller or larger groups will work. If vocalist(s) join in they can use any vowel and consonant sounds they like.

Patterns are to be played consecutively with each performer having the freedom to determine how many times he or she will repeat each pattern before moving on to the next. There is no fixed rule as to the number of repetitions a pattern may have, however, since performances normally average between 45 minutes and an hour and a half, it can be assumed that one would repeat each pattern from somewhere between 45 seconds and a minute and a half or longer.

It is very important that performers listen very carefully to one another and this means occasionally to drop out and listen. As an ensemble, it is very desirable to play very softly as well as very loudly and to try to diminuendo and crescendo together.

Each pattern can be played in unison or canonically in any alignment with itself or with its neighboring patterns. One of the joys of IN C is the interaction of the players in polyrhythmic combinations that spontaneously arise between patterns. Some quite fantastic shapes will arise and disintegrate as the group moves through the piece when it is properly played.

It is important not to hurry from pattern to pattern but to stay on a pattern long enough to interlock with other patterns being played. As the performance progresses, performers should stay within 2 or 3 patterns of each other. It is important not to race too far ahead or to lag too far behind.

The ensemble can be aided by the means of an eighth note pulse played on the high c's of the piano or on a mallet instrument. It is also possible to use improvised percussion in strict rhythm (drum set, cymbals, bells, etc.), if it is carefully done and doesn't overpower the ensemble. All performers must play strictly in rhythm and it is essential that everyone play each pattern carefully. It is advised to rehearse patterns in unison before attempting to play the piece, to determine that everyone is playing correctly.

page 2

The tempo is left to the discretion of the performers, obviously not too slow, but not faster than performers can comfortably play.

It is important to think of patterns periodically so that when you are resting you are conscious of the larger periodic composite accents that are sounding, and when you re-enter you are aware of what effect your entrance will have on the music's flow.

The group should aim to merge into a unison at least once or twice during the performance. At the same time, if the players seem to be consistently too much in the same alignment of a pattern, they should try shifting their alignment by an eighth note or quarter note with what's going on in the rest of the ensemble.

It is OK to transpose patterns by an octave, especially to transpose up. Transposing down by octaves works best on the patterns containing notes of long durations. Augmentation of rhythmic values can also be effective.

If for some reason a pattern can't be played, the performer should omit it and go on.

Instruments can be amplified if desired. Electronic keyboards are welcome also.

IN C is ended in this way: when each performer arrives at figure #53, he or she stays on it until the entire ensemble has arrived there. The group then makes a large crescendo and diminuendo a few times and each player drops out as he or she wishes.

[illegible]

C.2 Coded Score

This is a short excerpt from the score as coded for the agent system:

Listing C.1: Short excerpt from In C as coded for the agent system

```
( Piece           InC
  BPM:  120
  ( Section main
    RepeatUntil:      followLocus
    ( Section       s0
      Derives:      baseSection
      Length:       0.5
    )

    ( Section       s1
      Derives:      baseSection
      Length:       3
      ( Channel Notes
        ( Gracenote: C, 0)
        ( Note: E, 0.0, 1.0 )
        ( Gracenote: C, 1)
        ( Note: E, 1.0, 1.0 )
        ( Gracenote: C, 2)
        ( Note: E, 2.0, 1.0)
      )
    )

    ( Section       s2
      Derives:      baseSection
      Length:       2
      ( Channel Notes
        ( Gracenote: C, 0)
        ( Note: E, 0, 0.4 )
        ( Note: F, 0.5, 0.4 )
        ( Note: E, 1, 0.7)
      )
    )
```

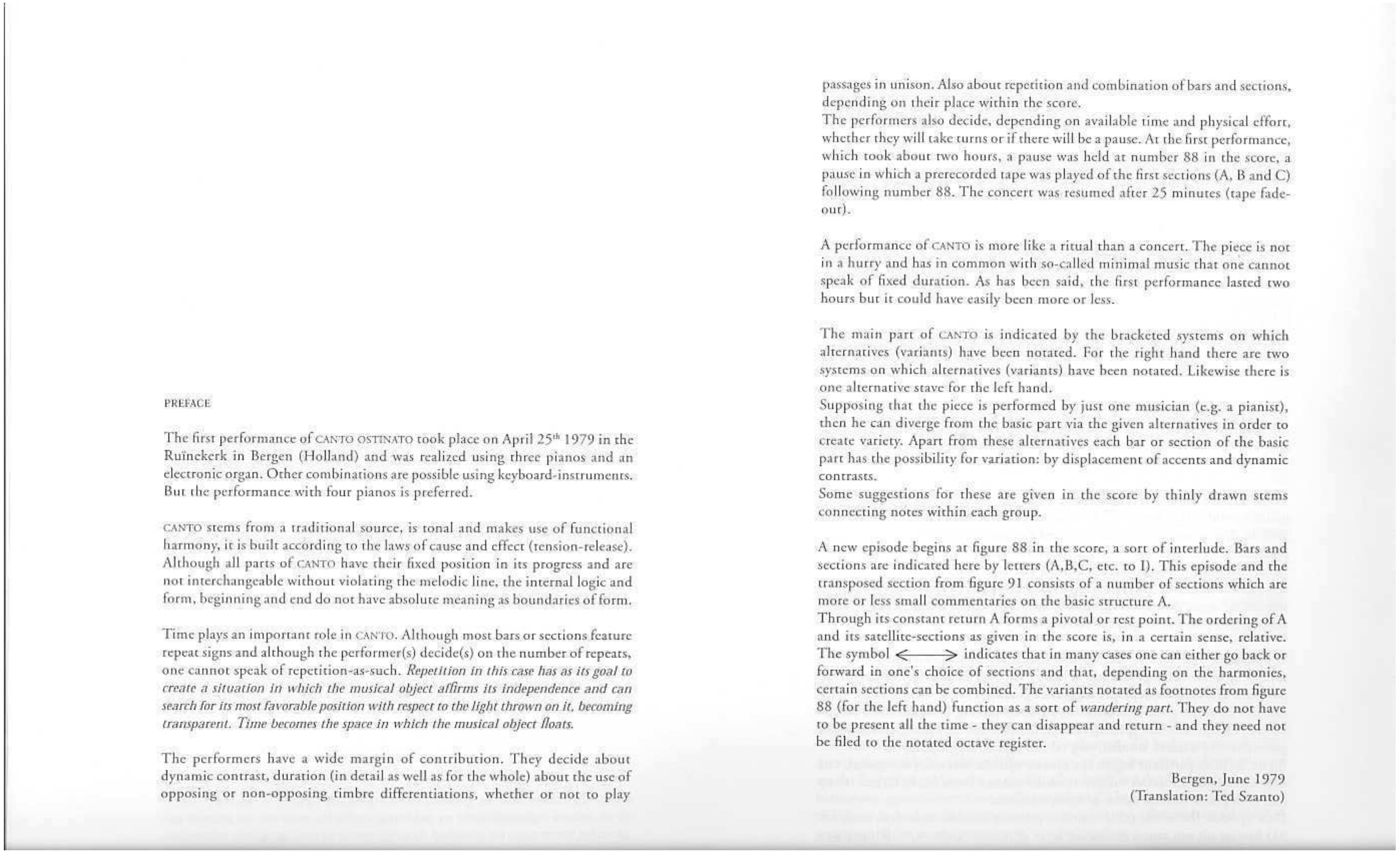
Appendix D

Canto Ostinato

D.1 Excerpts from the score

The following pages show:

- The full score notes
- The first excerpt used, from section 2 to section 11 (3 pages)
- the first page from each of the next two excerpts (sections 69 and 88)



PREFACE

The first performance of CANTO OSTINATO took place on April 25th 1979 in the Ruïnekerk in Bergen (Holland) and was realized using three pianos and an electronic organ. Other combinations are possible using keyboard-instruments. But the performance with four pianos is preferred.

CANTO stems from a traditional source, is tonal and makes use of functional harmony, it is built according to the laws of cause and effect (tension-release). Although all parts of CANTO have their fixed position in its progress and are not interchangeable without violating the melodic line, the internal logic and form, beginning and end do not have absolute meaning as boundaries of form.

Time plays an important role in CANTO. Although most bars or sections feature repeat signs and although the performer(s) decide(s) on the number of repeats, one cannot speak of repetition-as-such. *Repetition in this case has as its goal to create a situation in which the musical object affirms its independence and can search for its most favorable position with respect to the light thrown on it, becoming transparent. Time becomes the space in which the musical object floats.*

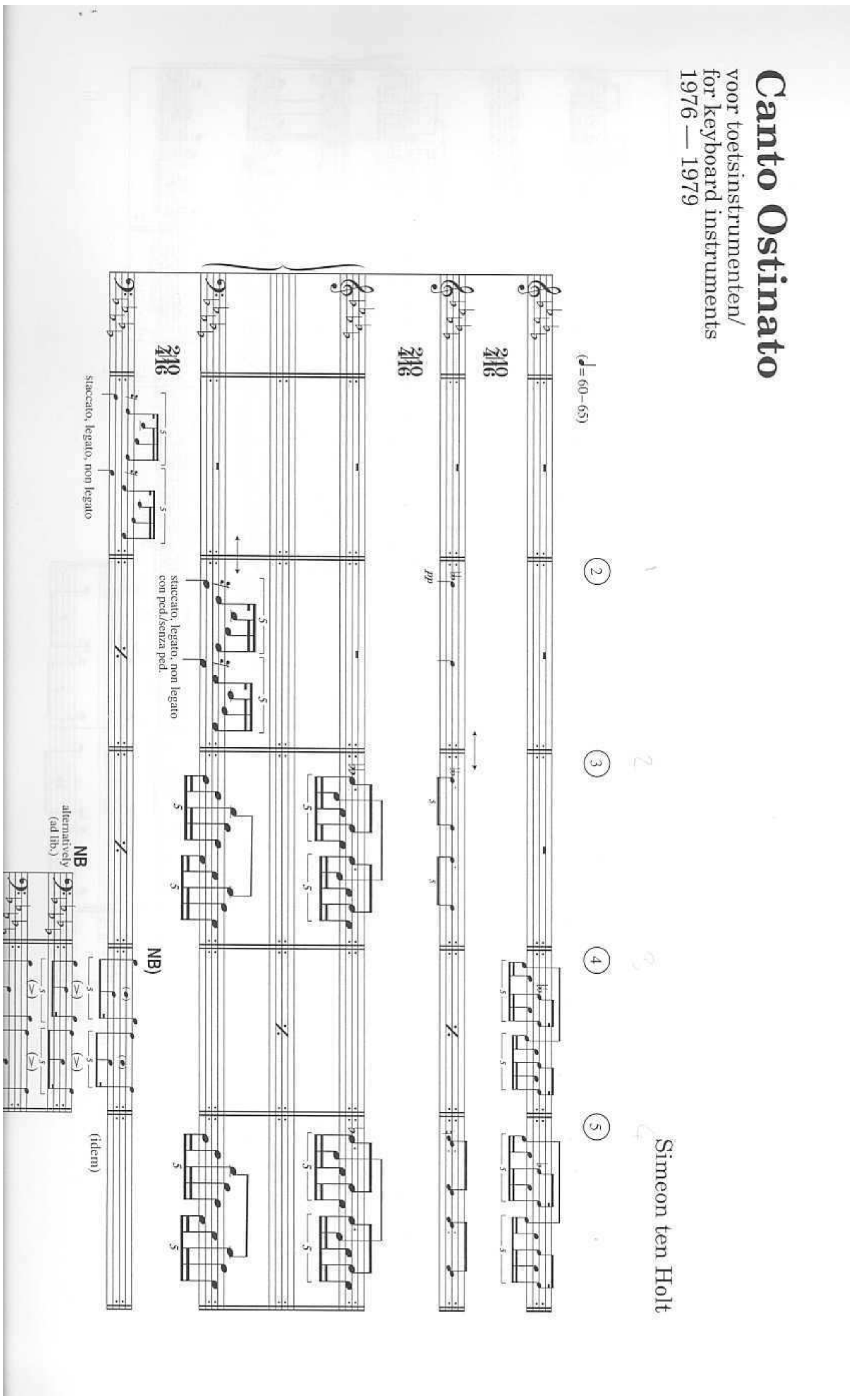
The performers have a wide margin of contribution. They decide about dynamic contrast, duration (in detail as well as for the whole) about the use of opposing or non-opposing timbre differentiations, whether or not to play passages in unison. Also about repetition and combination of bars and sections, depending on their place within the score.
The performers also decide, depending on available time and physical effort, whether they will take turns or if there will be a pause. At the first performance, which took about two hours, a pause was held at number 88 in the score, a pause in which a prerecorded tape was played of the first sections (A, B and C) following number 88. The concert was resumed after 25 minutes (tape fade-out).

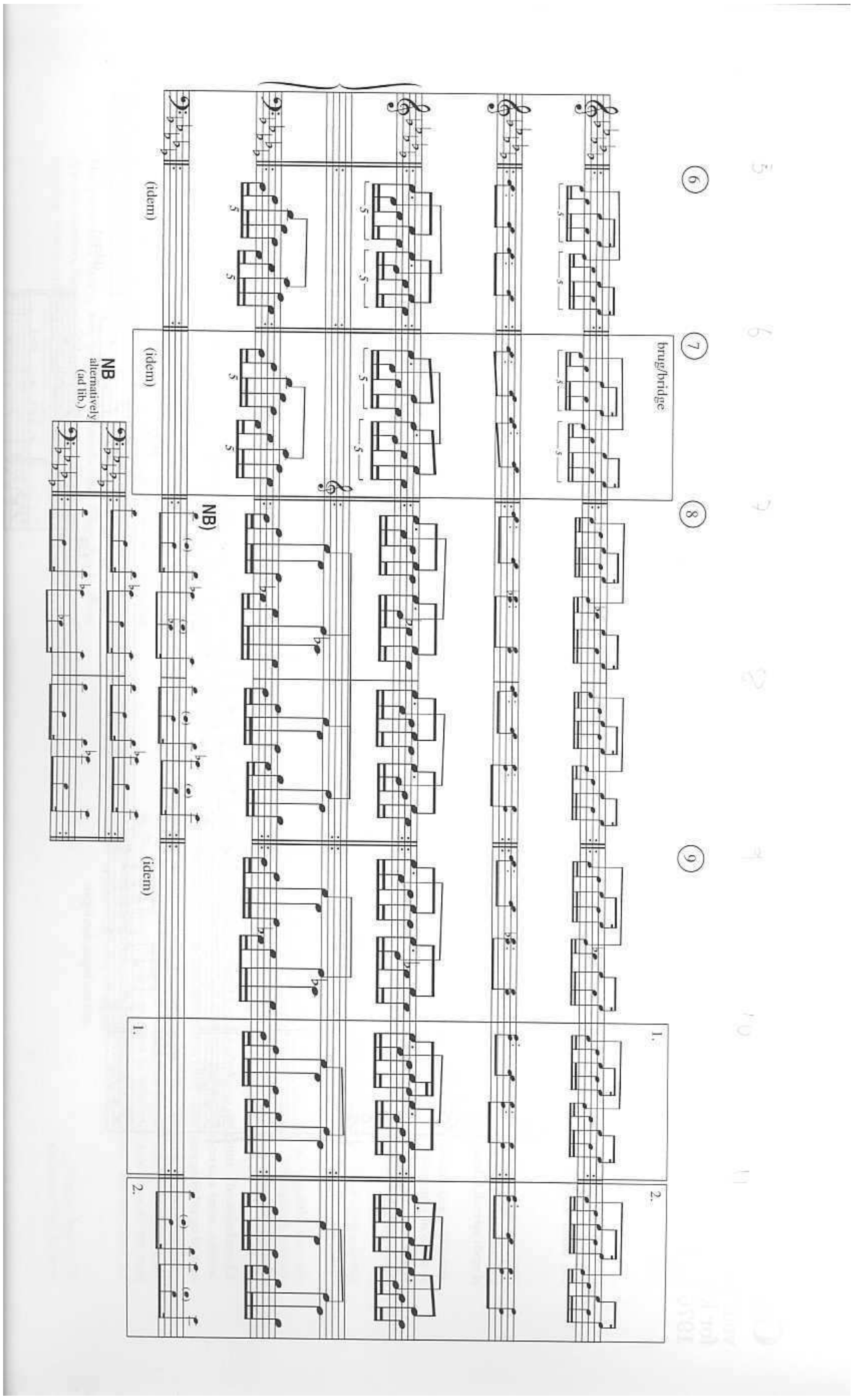
A performance of CANTO is more like a ritual than a concert. The piece is not in a hurry and has in common with so-called minimal music that one cannot speak of fixed duration. As has been said, the first performance lasted two hours but it could have easily been more or less.

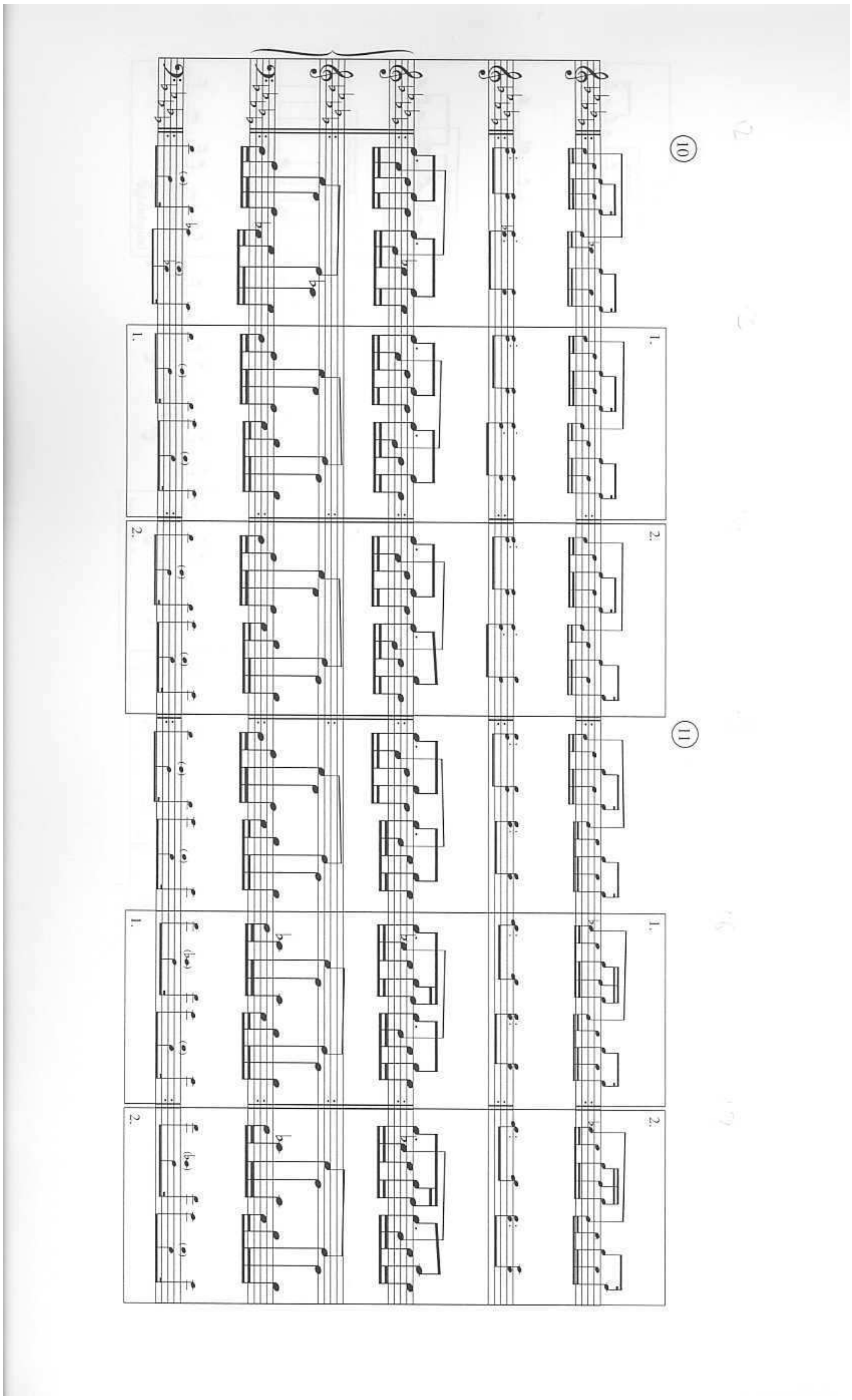
The main part of CANTO is indicated by the bracketed systems on which alternatives (variants) have been notated. For the right hand there are two systems on which alternatives (variants) have been notated. Likewise there is one alternative stave for the left hand.
Supposing that the piece is performed by just one musician (e.g. a pianist), then he can diverge from the basic part via the given alternatives in order to create variety. Apart from these alternatives each bar or section of the basic part has the possibility for variation: by displacement of accents and dynamic contrasts.
Some suggestions for these are given in the score by thinly drawn stems connecting notes within each group.

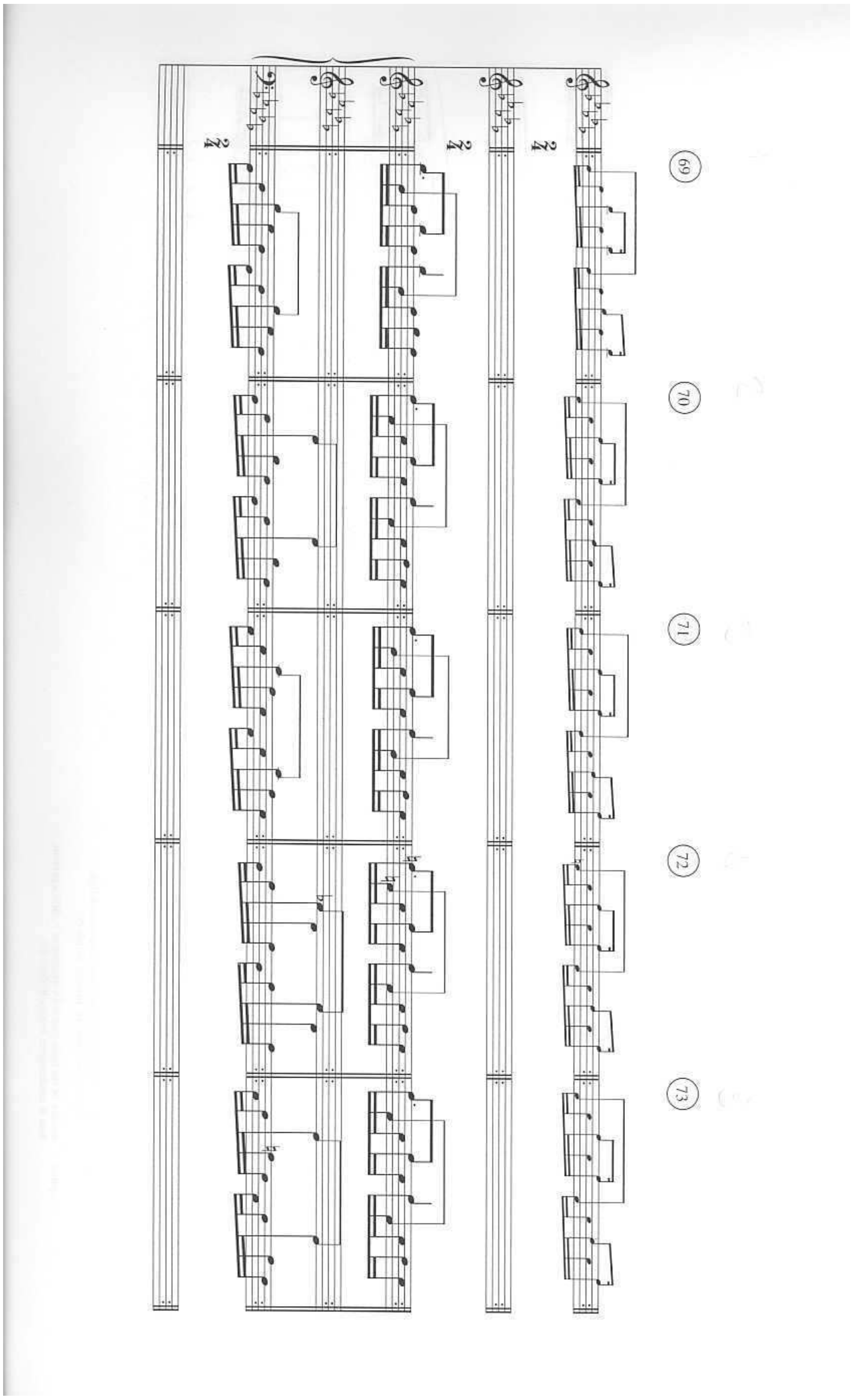
A new episode begins at figure 88 in the score, a sort of interlude. Bars and sections are indicated here by letters (A,B,C, etc. to I). This episode and the transposed section from figure 91 consists of a number of sections which are more or less small commentaries on the basic structure A.
Through its constant return A forms a pivotal or rest point. The ordering of A and its satellite-sections as given in the score is, in a certain sense, relative. The symbol $\longleftrightarrow$ indicates that in many cases one can either go back or forward in one's choice of sections and that, depending on the harmonies, certain sections can be combined. The variants notated as footnotes from figure 88 (for the left hand) function as a sort of *wandering part*. They do not have to be present all the time - they can disappear and return - and they need not be filed to the notated octave register.

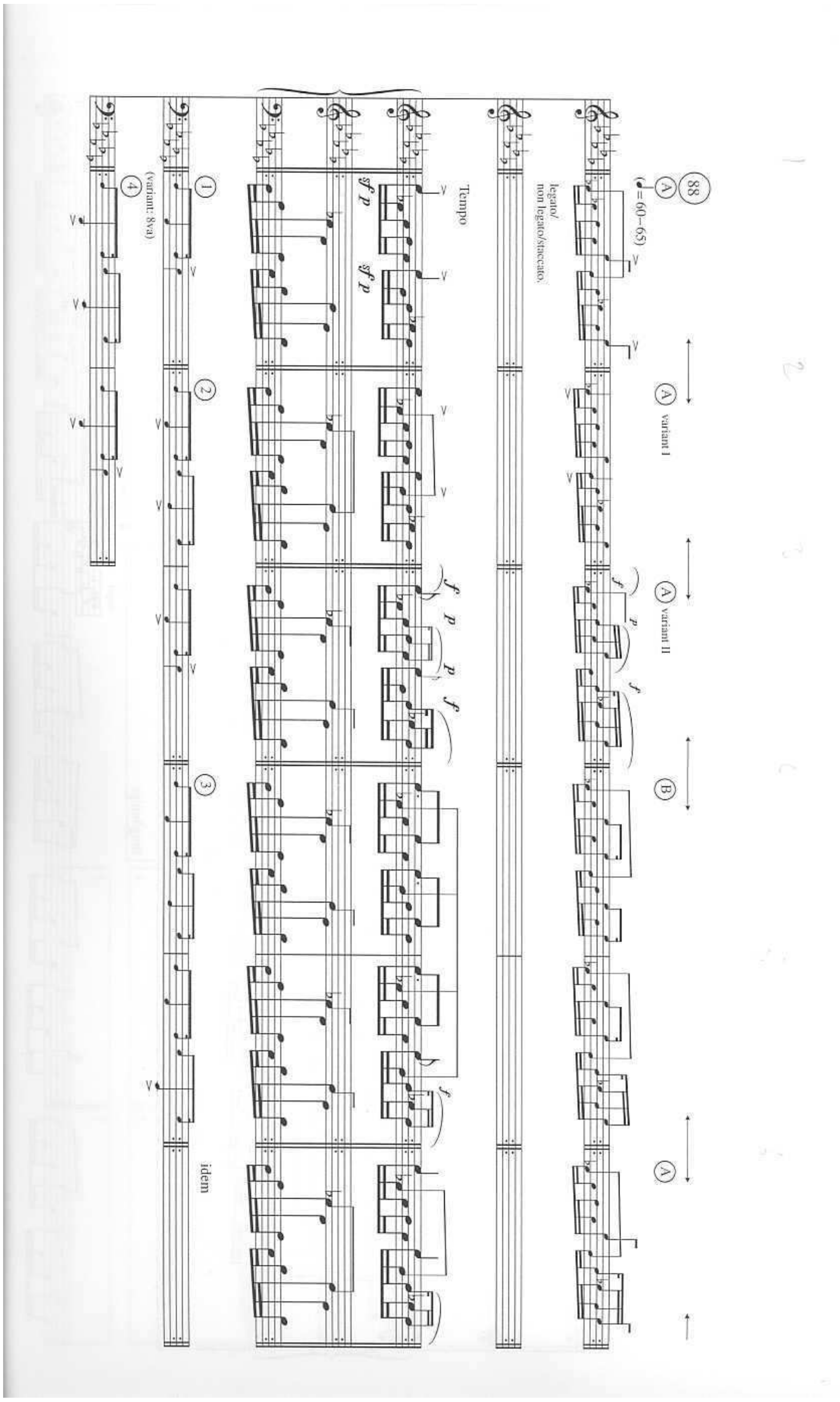
Bergen, June 1979
(Translation: Ted Szanto)


Canto Ostinato
voor toetsinstrumenten/
for keyboard instruments
1976 — 1979
Simeon ten Holt
(♩= 60–65)
staccato, legato, non legato
2
pp
staccato, legato, non legato
con ped./senza ped.
3
NB
alternatively
(ad lib.)
4
NB)
5
(idem)


6
7
brug/bridge
8
9
(idem)
(idem)
NB)
(idem)
NB
alternatively
(ad lib.)
1.
2.


10
11
1.
2.
1.
2.


69
70
71
72
73


88
A
(♩= 60–65)
legato/
non legato/staccato
Tempo
A variant I
A variant II
B
A
1
2
3
4
(variant: 8va)
idem
sf p
f p

D.2 Coded Score

This is a short excerpt from the score as coded for the agent system:

Listing D.1: Short excerpt from Canto Ostinato as coded for the agent system

```
( Piece CantoOstinato1-9
  BPM: 54.0
  Quantisation: 5
  ( Section main
    ( Section s2
      Repeats:    3
      Length:     2.00
      (Channel NotesLH
        ( Note: Bb1, 0.00, 0.20)
        ( Note: F2, 0.20, 0.20)
        ( Note: C#3, 0.40, 0.20)
        ( Note: Bb2, 0.60, 0.20)
        ( Note: F2, 0.80, 0.20)
        ( Note: Ab1, 1.00, 0.20)
        ( Note: Eb2, 1.20, 0.20)
        ( Note: C3, 1.40, 0.20)
        ( Note: Ab2, 1.60, 0.20)
        ( Note: Eb2, 1.80, 0.20)
      )
      (Channel NotesRH
      )
    )
    ( Section s3
      Length:     2.00
      Repeats:    3
      (Channel NotesLH
        ( Note: Bb1, 0.00, 0.20)
        ( Note: F2, 0.20, 0.20)
        ( Note: C#3, 0.40, 0.20)
        ( Note: Bb2, 0.60, 0.20)
        ( Note: F2, 0.80, 0.20)
        ( Note: Ab1, 1.00, 0.20)
        ( Note: Eb2, 1.20, 0.20)
        ( Note: C3, 1.40, 0.20)
        ( Note: Ab2, 1.60, 0.20)
        ( Note: Eb2, 1.80, 0.20)
      )
      (Channel NotesRH
        ( Note: A3, 0.00, 0.20)
        ( Note: C#3, 0.20, 0.20)
        ( Note: F3, 0.40, 0.20)
        ( Note: A3, 0.60, 0.20)
        ( Note: F3, 0.80, 0.20)
        ( Note: A3, 1.00, 0.20)
```

```
            ( Note: Eb3, 1.20, 0.20)
            ( Note: Ab3, 1.40, 0.20)
            ( Note: A3, 1.60, 0.20)
            ( Note: Ab3, 1.80, 0.20)
        )
    )
```

Appendix E

Example Questionnaire: Piano Duet Interaction Survey

E.1 Instructions

You will be asked to fill out a series of questions which describe your musical activities and competence, and some preferences about playing with other musicians. You will then be asked to play a series of excerpts with an unseen partner, which you will rate for several qualities. Your partner will not be able to see your ratings. Finally, you will be asked to fill out a few questions about your overall experience.

Since this is a long running experiment, we are not able to debrief you immediately. However, if you give your email address to the investigator, you will receive full details once the experiment is complete (although no participant ratings will be shared at any time).

E.1.1 Playing Instructions

The excerpts you will play are from Canto Ostinato, by Simeon ten Holt. The score for these excerpts has been attached, along with the score notes. You are asked to familiarise yourself with the excerpts beforehand, so that you can play one part at once without difficulty. The score, despite specifying the notes to be played, is open to a large degree of interpretation, with regard to dynamics, articulation, patterns of accents etc. - you are encouraged to make full use of this freedom in building an interpretation with your partner.

For each excerpt, you will be given a range of sections to play, and be told which "hand" you will be playing. For each section, you may choose one of the range of variations apropriate to that hand, in whatever manner you see fit. The score does not specify a number of repeats for each section - this is an arrangement arrived at by the performers. In some excerpts you will be told how many times to repeat each section - as if you had previously rehearsed and decided on this number. In other excerpts this may be unspecified, and it will be up to you and your partner to respond to each other's actions in an apropriate manner.

You will both play wearing headphones. You will hear your playing, and that of a partner in your headphones, along with a metronome. You are expected to follow the metronome as closely as possible, but this does not mean you must play completely mechanically.

Each excerpt will also specify who is "in control". If you are in control, then it is up to you to determine the course of the excerpt, and your partner will be expected to be responsive to your actions. Similarly, if your partner is in control, you should react to their playing apropriately. Some excerpts will specify that control should be passed between partners in a fluid manner, so that sometimes you are reacting to your partner, and sometimes your partner reacts to you.

E.2 Initial Questions

E.2.1 Personal Information

The following questions concern your abilities, competences and experiences with relevant types of music. You do not have to answer any questions you are not comfortable with.

What is your gender?

FEMALE
MALE

Are you left or righthanded?

Right handed
Left handed
Ambidextrous

How old are you?

less than 18
18-25
25-35
35-50
50-70
over 70

E.2.2 Musical Competences

These questions look at how much you play music, and how familiar you are with the particular piece and style to be used in this experiment. Some of the questions ask you to rate yourself on a scale; please do not try to be modest.

How many years have you been playing a musical instrument (Please cirle one)

less than 6 months
6 months to 1 year
1 - 2 years
2 - 3 years
3 - 5 years
more than 5 years

How many years have you been playing piano

less than 6 months
6 months to 1 year
1 - 2 years
2 - 3 years
3 - 5 years
more than 5 years

How many hours a week (on average) do you practise the piano less than 1

1 - 2

2 - 4

4 - 10

more than 10

How much do you listen to minimalist music

not at all __ __ __ __ __ __ __ very often

How often do you play minimalist music?

not at all __ __ __ __ __ __ __ very often

How familiar are you with Canto Ostinato

not at all __ __ __ __ __ __ __ played it in concert

Is the piano your main instrument?

no

yes

After briefly practising the piece, how easy do you find it to play the correct notes, and follow the correct structure?

very difficult __ __ __ __ __ __ __ easy

After briefly practising the piece, how well do you think you will be able to play these excerpts with another person?

very badly	__	__	__	__	__	__	__	very well

E.2.3 Playing with others

The following questions concern your attitude to playing music with others. Please tick a value which represents how important you find these qualities in a musical partner

How often do you play music with at least one other person

rarely	__	__	__	__	__	__	__	every day

Your partner is prepared follow instructions

not important	__	__	__	__	__	__	__	very important

Your partner pays attention to your playing

not important	__	__	__	__	__	__	__	very important

Your partner uses expressive articulation while playing

not important	__	__	__	__	__	__	__	very important

Your partner has a musical sense of time

not important	__	__	__	__	__	__	__	very important

Your partner introduces new ideas

not important	__	__	__	__	__	__	__	very important

Your partner can stay in time

not important	__	__	__	__	__	__	__	very important

Your partner makes use of dynamics

not important	__	__	__	__	__	__	__	very important

Your partner makes suprising responses

not important	__	__	__	__	__	__	__	very important

E.3 Excerpts

You will now be given the details for several excerpts. After you have played each excerpt, you are asked to fill out the corresponding section of the sheet, and then indicate to the experimenter that you are ready to move on.

If you are unsure about a particular question, a central choice indicates either that you are unsure, or that you would give an "average" score for that question.

E.3.1 Interaction 1

Excerpt	1-9
Hand	RH

How would you rate your partner's competence based on the excerpt you just played

low __ __ __ __ __ __ __ high

Did your partner use dynamics expressively

no use of dynamics __ __ __ __ __ __ __ very expressive dynamics

Did you feel the other participant was paying attention to your playing

none at all __ __ __ __ __ __ __ complete attention

Did your partner introduce new ideas

no new ideas __ __ __ __ __ __ __ many new ideas

Did the other participant suprise you with their responses

not suprising	__	__	__	__	__	__	__	very suprising

Was your partner playing live, or were they pre-recorded?

pre-recorded	__	__	__	__	__	__	__	live

Did your partner vary their timing in a musical manner

no musical timing	__	__	__	__	__	__	__	very musical timing

Would you choose to play with your partner again based on the excerpt you just played?

avoid playing with partner	__	__	__	__	__	__	__	choose to play with partner

Did you and your partner stay in time

out of time	__	__	__	__	__	__	__	completely in time

Did you enjoy playing the excerpt you just played?

not enjoyable	__	__	__	__	__	__	__	highly enjoyable

Did your partner articulate phrases expressively

no expressive articulation	__	__	__	__	__	__	__	very expressive articulation

Did your partner take the apropriate role (as indicated in the instructions)

inapropriate role	__	__	__	__	__	__	__	apropriate role

(A similar section is now repeated for each short interaction)

E.4 Post Questions

Finally we would like to get some feedback on how you rate your participation in this experiment, and how you felt about the experiment as a whole

How would you rate your playing today relative to your general level of competence

very poor __ __ __ __ __ __ __ very good

Did you feel you had the opportunity to play to your standard, or were you restricted by the setup of the experiment

very restricted __ __ __ __ __ __ __ unrestricted

How difficult did you find it to play with a partner you could not see?

very hard __ __ __ __ __ __ __ not difficult

How difficult did you find it to play with the metronome

very hard __ __ __ __ __ __ __ not difficult

How well did you feel you could differentiate between the different partners or experimental setups

very poorly __ __ __ __ __ __ __ very well

Was there anything which was particularly intrusive, distressing or annoying about the experiment and the environment?

____

____

____

____

Are there any suggestions you would like to make, or general thoughts about the experiment?

____

____

____

____

Bibliography

AR Addessi and F. Pachet. Young children confronting the Continuator, an interactive reflective musical system. *Musicae Scientiae. Special Issue*, 2006.

Àlvaro Barbosa. Displaced soundscapes: a survey of networked systems for music and sonic art creation. *Leonardo Music Journal*, 13:53–9, 2003.

Torsten Anders, Christina Anagnostopoulou, and Michael Alcorn. Strasheela: Design and usage of a music composition environment based on the oz programming model. In P. Van Roy, editor, *MOZ 2004, LNCS 3389*, pages 277–291. Springer-Verlag, 2004.

Josep Lluís Arcos and Ramon López de Mántaras. An interactive case-based reasoning approach for generating expressive music. *Applied Intelligence*, 14:115–129, 2001.

Katie Atkinson, Trevor Bench-Capon, and Peter Mcburney. A dialogue game protocol for multi-agent argument over proposals for action. *Autonomous Agents and Multi-Agent Systems*, 11(2):153–171, 2005.

J.J. Aucouturier and F. Pachet. Ringomatic: a realtime interactive drummer using constraint-satisfaction and drum sound descriptors. *Proceedings of the International Conference on Music Information Retrieval*, 2005.

John Langshaw Austin. *How to do things with words*. Clarendon Press, 1962.

F. Baader. *The Description Logic Handbook: theory, implementation, and applications*. Cambridge University Press, 2003.

Alvaro Barbosa. Public sound objects: a shared environment for networked music practise on the web. *Organised Sound*, 10(3), 2005.

Marco Beghilli. Performative musical acts: The verdian achievement. In *Musical Signification*. Mouton de Gruyter, 1995.

Bernard Bel. Migrating Musical Concepts: An Overview of the Bol Processor. *Computer Music Journal*, 22(2):56–64, 1998.

Bruce Ellis Benson. *The Improvisation of Musical Dialogue*. Cambridge University Press, 2003. ISBN 0-521-00932-4.

Al Biles. Interactive GenJam: Integrating real-time performance with a genetic algorithm. *Proc. 1998 Intl. Computer Music Conf.(ICMC-98)*, 1998.

Al Biles. Life with GenJam: interacting with a musical IGA. *IEEE International Conference on Systems, Man, and Cybernetics*, 3, 1999.

Tim Blackwell. Swarm music: improvised music with multi-swarms. In *Proceedings of the 2003 AISB symposium on AI and Creativity in Arts and Science, Aberystwyth*, pages 41–9, 2003.

Tim Blackwell and Michael Young. Self-organised music. *Organised Sound*, 9(2): 123–136, 2004.

Margaret A. Boden. Creativity and artificial intelligence. *Artificial Intelligence*, 103 (1):347–356, 1998.

Ollie Bown. The extended importance of the social creation of value in evolutionary processes: A proposed model. In *Computational Creativity Workshop, ECAI, Riva Del Garda*, 2006.

G.M. Breakwell. *Research Methods in Psychology*. Sage Pubns, 2006.

Alan Bundy. What kind of field is AI. In *The foundation of artificial intelligence—a sourcebook*, pages 215–222. Cambridge University Press, New York, NY, USA, 1990. ISBN 0-521-35944-9.

Elaine Chew, Alexander Sawchuk, Carley Tanoue, and Roger Zimmermann. Segmental Tempo Analysis of Performances in User-Centered Experiments in the Distributed Immersive Performance Project. In *Proceedings of the Sound and Music Computing Conference*, Salerno, Italy, November 24-26 2005. URL `http://www.smc05.unisa.it`.

Eric Clarke. Expression in performance: Generativity, perception and semiosis. In J. Rink, editor, *The Practice of Performance: Studies in Musical Interpretation*, pages 21–54. Cambridge Universtiy Press, 1995.

Eric F. Clarke. Generative principles in music performance. In John A. Sloboda, editor, *generative processes in music*, pages 1–26. Oxford University Press, 1988.

J. Cohen. Quantitative methods in psychology: A power primer. *Psychological Bulletin*, 112(1):155–159, 1992.

J. Cohen. *Statistical Power Analysis for the Behavioral Sciencies*. Lawrence Erlbaum, 1988.

J. Cohen, P. Cohen, S.G. West, and L.S. Aiken. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. Lawrence Erlbaum, 2002.

D. Cope. *Computers and musical style*. AR Editions, Inc. Madison, WI, USA, 1991.

D. Cope. *Experiments in musical intelligence*. AR Editions, Inc. Madison, WI, USA, 1996.

David Cope. *Virtual Music*. MIT Press, 2001.

E. Coutinho, M. Gimenes, J.M. Martins, and E.R. Miranda. Computational Musicology: An Artificial Life Approach. *Proceedings of the 2nd Portuguese Workshop on Artificial Life and Evolutionary Algorithms Workshop, Covilha (Portugal), Springer Verlag*, 2005.

Kenny R. Coventry and Tim Blackwell. Pragmatics in language and music. In Matt Smith, Alan Smaill, and Geraint A. Wiggins, editors, *Music Education: An Artificial Intelligence Approach*, Workshops in Computing, pages 123–142. Springer, 1994.

Ian Cross. Music as biocultural phenomenon. *Annals of the New York Academy of Sciences*, 999:106–111, 2003. URL `http://www-ext.mus.cam.ac.uk/~ic108/PDF/IRMCNYAS2003.PDF`.

R. Dannenberg. Listening to “Naima”: An Automated Structural Analysis of Music from Recorded Audio. *Proceedings of the 2002 International Computer Music Conference, Göteborg*, pages 28–34, 2002.

R.B. Dannenberg and N. Hu. Discovering Musical Structure in Audio Recordings. *International Conference on Music and Artificial Intelligence, LNCS Vol. 2445*, 2002.

Miles Davis. *Big Fun (reissue)*, chapter Little Blue Frog. Columbia/Legacy, 2000. Big Fun was originally published in 1969, but Little Blue Frog is taken from the Bitches Brew sessions of the same year, and added on the Columbia reissue.

E.A. Emerson. Temporal and modal logic. *Handbook of theoretical computer science (vol. B): formal models and semantics table of contents*, pages 997–1072, 1991.

Fabio De Felice, Fabio Abbattista, and Francesco Scagliola. Genorchestra: An interactive evolutionary agent for musical composition. In Celestino Soddu, editor, *Proceedings of the 5th International Conference GA2002*, Politecnico di Milano University, Italy, 2002.

FIPA. FIPA communicative act library specification. Foundation for Intelligent Physical Agents, `www.fipa.org/specs/fipa00037`, 2002.

FIPA Specification. http://www.fipa.org/repository/bysubject.html.

Joseph Rukshan Fonseka. Musical agents. Honours thesis, Monash University, 2000.

A. Forte and S.E. Gilbert. *Introduction to Schenkerian Analysis*. Norton, 1982.

A. Friberg, V. Colombo, L. Frydén, and J. Sundberg. Generating musical performances with director musices. *Computer Music Journal*, 15:23–29, 2000.

A. Gabrielsson and P.N. Juslin. Emotional Expression in Music Performance: Between the Performer's Intention and the Listener's Experience. *Psychology of Music*, 24 (1):68, 1996.

Alf Gabrielsson. Music performance research at the millennium. *Psychology of Music*, 31:221–272, 2003.

Alf Gabrielsson. Once again: The theme from Mozart's piano sonata in A major: A comparison of five performances. *Action and Perception in Rhythm and Music*, 55: 81–103, 1987.

Alf Gabrielsson. The performance of music. In D. Deutsch, editor, *The Psychology of Music*, pages 501–602. Academic Press, 1999.

P. Gannon. Band-in-a-Box. PG Music. Software Package, 1991. URL `http://www.pgmusic.com/`.

B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag New York, Inc. Secaucus, NJ, USA, 1997.

Stan Godlovitch. *Musical performance: a philosophical study*. Routledge (UK), 1998.

Patrick Greussay. Exposition ou exploration: Graphes beethoveniens. In Christian Bourgois, editor, *Quoi, quand, comment: la recherche musicale*. IRCAM, 1985.

H. Grice. Logic and conversation. In P. Cole and J. Morgan, editors, *Syntax and Semantics*, volume 3: Speech Acts. New York: Academic Press, 1975.

H. P. Grice. Meaning. *Philosophical Review*, 66:377–88, 1957.

M. Harris, A. Smaill, and G. Wiggins. Representing music symbolically. In *IX Colloquio di Informatica Musicale*, Genoa, Italy, 1991.

M. Heidegger. *The Origin of the Work of Art*. University of Waterloo, 1963.

M. Henz, S. Lauer, and D. Zimmermann. Compoze - intention-based music composition through constraint programming. In *Tools with Artificial Intelligence, 1996., Proceedings Eighth IEEE International Conference on*. IEEE, 1996.

Andreas Herzig and Dominique Longin. A logic of intention with cooperation principles and with assertive speech acts as communication primitives. In *Proc. 1st Int. Joint Conf. on Autonomous Agent and Multi-Agent System (AAMAS 2002)*, pages 920–927, Bologna, 15-19 juillet 2002. ACM Press.

Ray Jackendoff. *Consciousness and the Computational Mind*. MIT Press, 1987.

Jade Guide. Jade programmer's guide. http://jade.tilab.com/doc/programmersguide.pdf.

PN Johnson-Laird. How Jazz Musicians Improvise. *Music Perception*, 19(3):415–442, 2002.

S. Jordà, M. Kaltenbrunner, G. Geiger, and R. Bencina. The reacTable*. *Proceedings of the International Computer Music Conference (ICMC 2005), Barcelona, Spain*, 2005.

Patrik N. Juslin. Communicating emotion in music performance: A review and a theoretical framework. In Patrik N. Juslin and John Sloboda, editors, *Music and Emotion: Theory and Research*, pages 305–33. Oxford University Press, 2001.

Patrik N. Juslin. Five facets of musical expression: a psychologist's perspective on music performance. *Psychology of Music*, 31(3):273–302, 2003.

Ajay Kapur, Ge Wang, Philip Davidson, and Perry Cook. Interactive network performance: a dream worth dreaming? *Organised Sound*, 10(3):209–219, 2005.

J. Kippen and B. Bel. Modeling Music with Grammars: Formal Language Representation in the Bol Processor. *Computer Representations and Models in Music*, pages 207–232, 1992.

KQML Spec. KQML Specification. http://www.cs.umbc.edu/kqml/kqmlspec/spec.html.

S. Krakowiak. What is middleware. This is an electronic document. Date of publication: January 2003. URL `http://middleware.objectweb.org/`.

LAM. Live algorithms for music. Website, read on September 24th 2007. URL `http://www.livealgorithms.org/`.

Leigh Landy, editor. *Organised Sound*, volume 10. Cambridge University Press, 2005.

Fred Lerdahl. Cognitive constraints on compositional systems. In John A. Sloboda, editor, *generative processes in music*, pages 231–259. Oxford University Press, 1988.

Fred Lerdahl and Ray Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, 1983.

Stephen C. Levinson. *Pragmatics*. Cambridge University Press, 1983.

R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 140, 1932.

H.C. Longuet-Higgins. *Mental processes*. MIT Press Cambridge, Mass, 1987.

J. M. Martins and E. R. Miranda. Emergent rhythmic phrases in an a-life environment. In *Proceedings of the Workshop on Music and Artificial Life - European Conferecence of Artificial Life (ECAL)*, 2007.

G. Mazzola and G. Milmeister. Functors for Music: The Rubato Composer System. In *Proceedings of the ICMC*, pages 83–90, San Francisco: International Computer Music Association, 2006.

Guerino Mazzola. *The Topos of Music – Geometric Logic of Concepts, Theory and Performance*. Birkhäuser Verlag, 2002.

MIDI Specification. Midi specifications. URL `http://www.midi.org/about-midi/specshome.shtml`. http://www.midi.org/about-midi/specshome.shtml.

Evenadro Manara Miletto, Marcelo Soares Pimenta, Rosa Maria Vicari, and Luciano Vargas Flores. Codes: a web-based environment for cooperative music prototyping. *Organised Sound*, 10(3):243–253, 2005.

Marvin Minsky. *The society of mind*. Simon & Schuster, Inc., 1986.

Marvin Minsky. Music, mind, and meaning. *Computer Music Journal*, 5(3):28–44, 1981. URL `http://web.media.mit.edu/~minsky/papers/MusicMindMeaning.html`.

E. R. Miranda. Emergent sound repertoires in virtual societies. *Computer Music Journal*, 26(2):77–90, 2002.

E.R. Miranda. On the evolution of music in a society of self-taught digital creatures. *Digital Creativity*, 14(1):29–42, 2003.

Charles Morris. *Foundations of the Theory of Signs*. Chicago, 1938.

D. S. Murray-Rust and A. Smaill. Musical acts and musical agents. In *Proceedings of the 5th Open Workshop of MUSICNETWORK: Integration of Music in Multimedia Applications*, 2005. http://homepages.inf.ed.ac.uk/s0239182/Murray-Rust_MusicalActs.pdf.

David Murray-Rust. Virtualatin - agent based percussive accompaniment. Master's thesis, University of Edinburgh, 2003. http://www.inf.ed.ac.uk/publications/thesis/online/IM030053.pdf.

David Murray-Rust, Alan Smaill, and Michael Edwards. MAMA: An architecture for interactive musical agents. In *Proceedings of ECAI 2006*, pages 36–40. IOS Press, 2006.

David S. Murray-Rust. Virtualatin: Towards a musical multi agent system. In *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications*, pages 17–22. IEEE CS Press, 2005.

E. Narmour. *Beyond Schenkerism: The Need for Alternatives in Music Analysis*. University of Chicago Press, 1977.

E. Narmour. *The Analysis and Cognition of Basic Melodic Structures: The Implication-Realization Model*. University of Chicago Press, 1990.

Maria Neurath. *Logical Empiricism at its Peak: Schlick, Carnap, and Neurath*. Routledge, 1996.

A. Omicini, A. Ricci, and M. Viroli. Agens Faber: Toward a Theory of Artefacts for MAS. *Electronic Notes in Theoretical Computer Science*, 150:21–36, 2006.

M.A. Orgun and W. Ma. An overview of temporal and modal logic programming. *Proceedings of the First International Conference on Temporal Logic*, pages 445–479, 1994.

F. Pachet. Rhythm as emerging structure. In *Proceedings of ICMC*, Berlin, 2000. ICMA.

F. Pachet. Enhancing individual creativity with an interactive reflective musical system. In I. Deliège and G. H. Wiggins, editors, *Musical Creativity: Current Research in Theory and Practice.* Hove: Psychology Press, 2004a.

F. Pachet. The MusES system: An environment for experimenting with knowledge representation techniques in tonal harmony. In *Proceedings of the 1st Brazilian Symposium on Computer Music, Caxambu, Minas Gerais, Brazil*, pages 195–201, 1994. URL `citeseer.nj.nec.com/148999.html`.

F. Pachet and A.R. Addessi. When children reflect on their own playing style: experiments with continuator and children. *Computers in Entertainment (CIE)*, 2(1): 14–14, 2004.

F. Pachet and P. Roy. Musical Harmonization with Constraints: A Survey. *Constraints*, 6(1):7–19, 2001.

F. Pachet, G. Ramalho, and J. Carrive. Representing temporal musical objects and reasoning in the MusES system. *Journal of New Music Research*, 5(3):252–275, 1996. URL `citeseer.nj.nec.com/pachet95representing.html`.

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

Francois Pachet. Beyond the cybernetic jam fantasy: The continuator. *IEEE Computers Graphics and Applications*, January/February 2004b. special issue on Emerging Technologies.

C. Palmer. Anatomy of a performance: Sources of musical expression. *Music Perception*, 13:433–53, 1996.

Marcus Pearce, David Meredith, and Geraint A. Wiggins. Motivations and methodologies for automation of the compositional process. *Musicae Scientiae*, 6(2):pp. 119–147, 2002. URL `citeseer.ist.psu.edu/pearce02motivations.html`.

M Pelz-Sherman. *A Framework for Performer Interactions in Western Improvised Contemporary Art Music*. PhD thesis, UC San Diego, 1998. URL `http://pelz-sherman.net/mpsdiss.pdf`.

J. Pressing. Cognitive processes in improvisation. In W. R. Crozier and A. J. Chapman, editors, *Cognitive processes in the perception of art*, pages 345–363. 1984.

Jeff Pressing. Improvisation: methods and models. In John A. Sloboda, editor, *Generative Processes in Music*, pages 192–178. Oxford University Press, 1988.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2007. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

Rafael Ramirez and Amaury Hazan. Modeling expressive music performance in jazz. In *Proceedings of the 18th Florida Artificial Intelligence Research Society Conference*, 2005.

C. Raphael. Music Plus One: A System for Expressive and Flexible Musical Accompaniment. *Proceedings of the ICMC, Havana, Cuba*, pages 159–162, 2001.

Robert Rowe. *Interactive Music Systems - Machine Listening and Composing*. MIT Press, 1993.

Ed Sarath. A new look at improvisation. *Journal of Music Theory*, 40(1):1–38, 1996.

J.R. Searle. *Intentionality: an essay in the philosophy of mind*. Cambridge University Press, 1983.

J.R. Searle. *Speech Acts*. Cambridge University Press, 1969.

F.A. Seddon. Modes of communication during jazz improvisation. *British Journal of Music Education*, 22(01):47–61, 2005.

John A. Sloboda. The communication of musical metre in piano performance. *Quarterly Journal of Experimental Psychology*, 35:377–96, 1983.

SW Smoliar. Parsing, structure, memory and affect. *Journal of New Music Research*, 24(1):21–33, 1995.

J.G. Snider and C.E. Osgood. *Semantic differential technique: A sourcebook*. Aldine Atherton, 1969.

R.C. Stalnaker. Pragmatics. *Synthese*, 22(1):272–289, 1970.

M. J. Steedman. The blues and the abstract truth: Music and mental models. In A. Garnham and J. Oakhill, editors, *Mental Models in Cognitive Science*, pages 305–318. Erlbaum, 1996.

I. Stravinsky. *Poetics of Music in the Form of Six Lessons*. Harvard University Press Cambridge, Mass, 1970.

D. Sudnow. *Ways of the Hand: The Organization of Improvised Conduct*. MIT Press, 1993.

Belinda Thom. BoB: An interactive improvisational music companion. In Carles Sierra, Maria Gini, and Jeffrey S. Rosenschein, editors, *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 309–316, Barcelona, Catalonia, Spain, 2000. ACM Press. URL `citeseer.nj.nec.com/thom00bob.html`.

N. P. M. Todd. The dynamics of dynamics. *Journal of the Acoustical Society of America*, 91:3540–3550, 1992.

C. Truchet and P. Codognet. Musical constraint satisfaction problems solved with adaptive search. *Soft Computing*, 8:633–640, 2004.

Leo Kazuhiro Ueda and Fabio Kon. Andante: A mobile musical agents infrastructure. In *Proceedings of the IX Brazilian Symposium on Computer Music*, pages 87–94, Campinas, Brazil, 2003.

M. Viroli and A. Ricci. Instructions-based semantics of agent mediated interaction. *Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004. Proceedings of the Third International Joint Conference on*, pages 102–109, 2004.

William F. Walker. A computer participant in musical improvisation. In *Proceedings of Conference on Human Factors in Computing Systems CHI97*, pages 123–130, 1997.

Christopher D. Walton. Multi-agent dialogue protocols. In *AI&M 2004, Eighth International Symposium on Artificial Intelligence and Mathematics*, 2004.

Gil Weinberg. The aesthetics, history and future challenges of interconnected music networks. In *Proceedings of the International Computer Music Association Conference*, pages 349–56, Göteborg, Sweeden, 2002.

G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.

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

Gerhard Widmer. Using AI and machine learning to study expressive music performance: project survey first report. *AI Communications*, 14(3):149–162, 2001.

Gerhard Widmer and Werner Goebl. Computational models of expressive music performance: The state of the art. *Journal of New Music Research*, 33(3):203–216, 2004.

Geraint Wiggins, Eduardo Miranda, Alan Smaill, and Mitch Harris. Surveying musical representation systems: A framework for evaluation. *Computer Music Journal*, 17 (3), 1993.

M. Wooldridge. *Reasoning about Rational Agents*. MIT, Cambridge, MA, 2002.

Michael J. Wooldridge. *An introduction to MultiAgent systems*. John Wiley and Sons, 2001.

Matthew Wright. Open sound control: an enabling technology for musical networking. *Organised Sound*, 10(3):193–200, 2005.

Rodolfo Daniel Wulfhorst, Lauro Nakayama, and Rosa Maria Vicari. A multiagent approach for musical interactive systems. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 584–591. ACM Press, 2003.

D. Zimmermann. Automatic composition of intention-based music for multimedia interfaces. In *Proceedings of First International Workshop on Intelligence of Multimedia Interfaces*, *IMMI1*, Edinburgh, Scotland, 1995a. AAAI Press.

D. Zimmermann. Exploiting models of musical structure for automatic intention-based composition of background music. In G. Widmer, editor, *Proceedings of the IJCAI'95 Workshop on Artificial Intelligence and Music*, Menlo Park, CA, 1995b. AAAI Press.