

**GENERATIVE MUSIC, COGNITIVE MODELLING,
AND COMPUTER-ASSISTED COMPOSITION
IN MUSICOG AND MANUSCORE**

by

James B. Maxwell

M.F.A., Simon Fraser University, 2001

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
Doctor of Philosophy

under Special Arrangements with
Dean of Graduate Studies
School for the Contemporary Arts and the
School of Interactive Art + Technology
Faculty of Communications, Art, and Technology

© James B. Maxwell 2014

SIMON FRASER UNIVERSITY

Summer 2014

All rights reserved.

However, in accordance with the *Copyright Act of Canada*, this work may be reproduced without authorization under the conditions for "Fair Dealing." Therefore, limited reproduction of this work for the purposes of private study, research, criticism, review and news reporting is likely to be in accordance with the law, particularly if cited appropriately.

Approval

Name: James B. Maxwell
Degree: Doctor of Philosophy
Title: *Generative Music, Cognitive Modelling,
And Computer-Assisted Composition In
MusiCog And ManuScore*

Examining Committee: Chair: Dean of Graduate Studies or designate

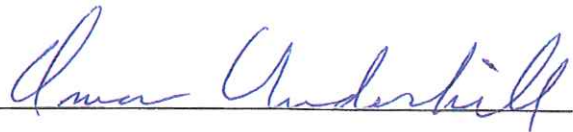
Dr. Philippe Pasquier
Senior Supervisor
Associate Professor, SFU
School of Interactive Art + Technology



Dr. Arne Eigenfeldt
Supervisor
Associate Professor, SFU
School for the Contemporary Arts



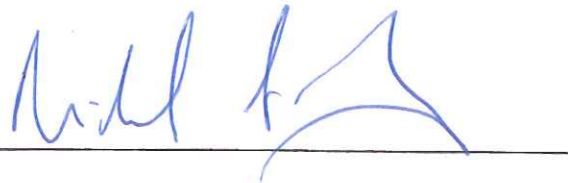
Owen Underhill
Supervisor
Professor, SFU
School for the Contemporary Arts



Kenneth Newby
Internal Examiner
Professor
University of the Fraser Valley



Dr. Michael Casey
External Examiner
Joint Professor
Music, Computer Science
Dartmouth College



Date Defended/Approved: August 07, 2014

Partial Copyright Licence



The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the non-exclusive, royalty-free right to include a digital copy of this thesis, project or extended essay[s] and associated supplemental files ("Work") (title[s] below) in Summit, the Institutional Research Repository at SFU. SFU may also make copies of the Work for purposes of a scholarly or research nature; for users of the SFU Library; or in response to a request from another library, or educational institution, on SFU's own behalf or for one of its users. Distribution may be in any form.

The author has further agreed that SFU may keep more than one copy of the Work for purposes of back-up and security; and that SFU may, without changing the content, translate, if technically possible, the Work to any medium or format for the purpose of preserving the Work and facilitating the exercise of SFU's rights under this licence.

It is understood that copying, publication, or public performance of the Work for commercial purposes shall not be allowed without the author's written permission.

While granting the above uses to SFU, the author retains copyright ownership and moral rights in the Work, and may deal with the copyright in the Work in any way consistent with the terms of this licence, including the right to change the Work for subsequent purposes, including editing and publishing the Work in whole or in part, and licensing the content to other parties as the author may desire.

The author represents and warrants that he/she has the right to grant the rights contained in this licence and that the Work does not, to the best of the author's knowledge, infringe upon anyone's copyright. The author has obtained written copyright permission, where required, for the use of any third-party copyrighted material contained in the Work. The author represents and warrants that the Work is his/her own original work and that he/she has not previously assigned or relinquished the rights conferred in this licence.

Simon Fraser University Library
Burnaby, British Columbia, Canada

revised Fall 2013

Abstract

Music composition is a complex, multi-modal human activity, engaging faculties of perception, memory, motor control, and cognition, and drawing on skills in abstract reasoning, problem solving, creativity, and aesthetic evaluation. For centuries musicians, theorists, mathematicians—and more recently computer scientists—have attempted to systematize composition, proposing various formal methods for combining sounds (or symbols representing sounds) into structures that might be considered musical. Many of these systems are grounded in the statistical modelling of existing music, or in the mathematical formalization of the underlying rules of music theory. This thesis presents a different approach, looking at music as a holistic phenomenon, arising from the integration of perceptual and cognitive capacities. The central contribution of this research is an integrated cognitive architecture (ICA) for symbolic music learning and generation called MusiCog. Inspired by previous ICAs, MusiCog features a modular design, implementing functions for perception, working memory, long-term memory, and production/composition. MusiCog’s perception and memory modules draw on established experimental research in the field of music psychology, integrating both existing and novel approaches to modelling perceptual phenomena like auditory stream segregation (polyphonic voice-separation) and melodic segmentation, as well as higher-level cognitive phenomena like “chunking” and hierarchical sequence learning. Through the integrated approach, MusiCog constructs a representation of music informed specifically by its perceptual and cognitive limitations. Thus, in a manner similar to human listeners, its knowledge of different musical works or styles is not equal or uniform, but is rather informed by the specific musical structure of the works themselves.

MusiCog’s production/composition module does not attempt to model explicit knowledge of music theory or composition. Rather, it proposes a “musically naïve” approach to composition, bound by the perceptual phenomena that inform its representation of musical

structure, and the cognitive constraints that inform its capacity to articulate its knowledge through novel compositional output.

This dissertation outlines the background research and ideas that inform MusiCog's design, presents the model in technical detail, and demonstrates through quantitative testing and practical music theoretical analysis the model's capacity for melodic style imitation when trained on musical corpora in a range of musical styles from the Western tradition. Strengths and limitations—both of the conceptual approach and the specific implementation—are discussed in the context of autonomous melodic generation and computer-assisted composition (CAC), and avenues for future research are presented. The integrated approach is shown to offer a viable path forward for the design and implementation of intelligent musical agents and interactive CAC systems.

To Martha Rose Grymaloski

“Music is the best means we have of digesting time.”

— *W. H. Auden*

Acknowledgments

I'd like to start by thanking my supervisor, Dr. Arne Eigenfeldt, for initially presenting the possibility that my interest in generative music and Computer-Assisted Composition might provide the foundation for a doctoral thesis. This work would not likely have taken the course it did without his encouragement. I'd also like to thank my senior supervisor, Dr. Philippe Pasquier, for his expert guidance and unrelenting attention to detail throughout this investigation. It has been an extremely challenging process for me to acquire the skills and knowledge necessary—in a field quite removed from my original area of expertise—to bring my research to this stage, and it would not likely have been possible without Philippe's exacting standards and demand for intellectual and academic rigour. Beyond their academic assistance, I'd also like to thank Arne and Philippe for providing me with invaluable opportunities to present my work both at home and abroad. Travelling to beautiful and fascinating places like Kobe, Padova, Copenhagen, and Ljubljana has been a “perk” of my doctoral studies that I will never forget. On that note, I should also thank SIAT's Desiré Nazareth for her admirable patience in handling my somewhat “creative” accounting practices, and for enabling me to placate the Gods of Credit in a timely manner on my return home. I also owe a debt of gratitude to the Social Sciences and Humanities Research Council of Canada (SSHRC), and the Dean of Graduate Studies office (through the Special Arrangements program) for their financial support. Credit for making the daily, weekly, and monthly hoop-jumping of the doctoral path manageable and eminently tolerable goes to the timely and well-tempered persistence of Sheilagh MacDonald. I'd also like to thank Prof. Owen Underhill for his quiet confidence in my progress throughout.

Doctoral research is more process than product, and processes are complex, multi-faceted things. As such, I would like to acknowledge the intellectual enthusiasm and generosity of my fellow researchers from the MAMAS Lab, who no doubt influenced my thinking in numerous and subtle ways. I hold a particular appreciation for the level-headed recalibration of perspective provided by my conversations with Graeme McCaig and Nicolas

Gonzalez Thomas. Also helpful were Nicolas' comments regarding my written description of the quantitative methods used in evaluating this work, for which an indirect thanks goes to Thomas M. Loughin from SFU's Department of Statistics and Actuarial Science. For proofreading assistance I owe congratulatory bottles of Visine™ to Teresa Connors, Claire French, and Charles Maxwell.

Of course, my biggest and warmest thanks goes to my friends and family for their patience and support throughout this process. In particular, to my better-than-better-half, Claire, who endured countless hours of fantastically incomprehensible speech and reckless flights of wild, intellectual abandon, punctuated by occasional swan-dives into mental turmoil and insomniac despair. I don't imagine she really slept through the routine periods of 5 a.m. coffee grinding, but she certainly never made it a point of contention. Her unwavering support, and absolute, unshakeable confidence in my abilities are a constant reminder of how incredibly fortunate I am. For inspiration, my brother Max and his partner-in-art-crime Hadley have always stood as towering figures in my mind, for the intellectual and conceptual rigour of their work, but also for their insistence that playfulness is more fundamental to art making than solemnity. The long nights wrapped in conversation, pop-music-historical taunting, and "Scopa!" (and warmed by more than a drop of Russian Standard) in Kreuzberg and Kreuzkölln, are some of the fondest memories of my doctoral years. Of course, I also have to thank my longest-standing editor, Chuck (a.k.a., "Pops"), who instilled in me a healthy disdain for the word "very," and an appreciation of the inherent value of a poetic turn of phrase, whatever the context. Finally, I owe my deepest and most heartfelt thanks to Dr. Martha Rose Grymaloski (a.k.a, "Ma"), who gave me the intellectual curiosity, mental tenacity, purposefulness, and self-discipline required to meet each new challenge with a taste for its achievement. Her spirit of inquisitiveness and her unconditional moral, artistic, and intellectual support sustain me in all my endeavours. Looking back, it's difficult to understand how I arrived at this point, with her taken away before I could reach the end, but I know she would have been immeasurably proud to have another doctor in the family. She is at once my deepest sorrow and my greatest inspiration.

Contents

Partial Copyright License	iii
Abstract	iv
Dedication	vi
Quotation	vii
Acknowledgments	viii
Contents	x
List of Tables	xiv
List of Figures	xv
List of Algorithms	xx
Preface	xxi
1 Introduction	1
1.1 Reformulating the Problem: A Holistic Approach	3
2 Composers and Authors Compose Compositions	6
2.1 Laske's Composition Theory	10
2.2 Composition Theory in Literature	13
2.3 Collins' Synthesis Process Model	18
2.4 Composition Theory and Generative Models	22
2.4.1 Composition by Prediction: The Markov Property	22
2.4.2 An Innate Faculty for Music: Generative Grammars	29

2.4.3	Iteration Toward Musical “Fitness:” Evolutionary Models	36
2.4.4	David Cope’s “Music Recombinance”	40
2.5	If Composers are Creative, What Does That Mean?	47
2.5.1	Boden’s Creative Magnitudes and Types	48
2.5.2	Schmidhuber’s Theory of “Compressor Improvement”	51
2.5.3	Graeme Ritchie: Quantifying Computational Creativity	52
3	Music Perception and Cognition	55
3.1	Echoic Memory	55
3.2	Auditory Stream Segregation	57
3.3	Short-Term Memory or “Working Memory”	59
3.4	Memory Optimization and “Chunking”	61
3.5	Cognitive Modelling	64
3.6	Cognitively-Grounded Music Generation Systems	64
3.6.1	The IDyOM Model	64
3.6.2	The Anticipatory Model of Cont et al.	66
3.6.3	A Deep Learning Approach	68
3.7	Integrated Cognitive Architectures	69
4	MusiCog: An Integrated Architecture	72
4.1	Music Descriptors	74
4.2	Processing Modules	79
4.2.1	Perception Module (PE)	79
4.2.2	Working Memory (WM)	90
4.2.3	Long-Term Memory (LTM)	98
4.2.4	The Production Module (PM)	110
4.2.5	Handling Rhythmic Information	122
4.3	Implementation Details	123
5	MusiCog in Practice	124
5.1	The Perception Module	124
5.1.1	PE Stream/Voice-Separation	124
5.1.2	PE Low-level Boundary Detection	127
5.1.3	PE Induction of Mode and Tonal Centre	129
5.2	The Working Memory Module	131

5.2.1	WM Cognitive Saliency and WM Retention	131
5.2.2	WM Chunking and Higher-level Segmentation	133
5.3	Long-Term Memory: Learning in the CM	137
5.4	Generation in the PM	142
5.4.1	Testing on the Folk Corpus	146
5.4.2	Imitation of more developed styles	156
6	Discussion: Autonomous Composition in MusiCog	165
6.1	Discussion of Test Results	165
6.2	Strengths of an Integrated Approach	170
6.3	Questions and Challenges	172
6.3.1	<i>Stylistically inconsistent rhythmic complexity and syncopation.</i>	172
6.3.2	<i>Inability to reliably produce tonal melodies, when trained on tonal materials.</i>	177
6.3.3	<i>Lack of formal repetition structure and symmetry.</i>	178
6.4	Creativity in MusiCog	181
7	ManuScore: Cognitively-Grounded Computer-Assisted Composition	183
7.1	Design Motivations Behind ManuScore	185
7.2	ManuScore Design & Features	186
7.2.1	An Open Musical Space	186
7.2.2	Note Entry in ManuScore.	189
7.2.3	Orchestration in ManuScore.	190
7.2.4	Sharing Staff Data with “Links”	193
7.3	A Composition Study Using ManuScore	195
7.4	A Listener Study	196
7.5	Study Results	197
7.6	ManuScore as a CAC Tool	199
7.7	Composition in ManuScore with MusiCog	202
8	Conclusion	204
8.1	Future Work	204
8.1.1	Improvements and Extensions to the PE	204
8.1.2	Improvements and Extensions to the PM	205
8.1.3	Modelling Attention	210

8.2 Final Statements	210
Bibliography	214
Appendix A Score Example 1: <i>experiri</i>	231
Appendix B Score Example 2: <i>factura</i>	242

List of Tables

2.1	A text with topics and nested subtopics produced by Hayes' "topic-elaboration" model.	16
2.2	A set of rewriting rules describing the "non-terminal" symbols in a generative grammar (from Nierhaus [177]).	30
2.3	A set of "terminal" symbols to be used with the generative grammar in Table 2.2 (from Nierhaus [177]).	30
2.4	A set of rewriting rules for Cope's SPEAC system (reproduced from Cope [52]).	44
4.1	Synthesized spectral weights used for calculating the harmonic support of adjacent and/or simultaneous MIDI pitches.	88
4.2	Estimating the mode using the chroma vector and a modal "masking" vector.	89
5.1	Learning at L_1 , L_2 , and L_3 for four different values of φ when trained on a corpus of jazz songs (top), and on the Bach BWV 1013 partita (bottom).	142
7.1	Audience evaluation of "engagement."	198
7.2	Audience evaluation of "directly human-composed."	199

List of Figures

2.1	Hayes' "topic-elaboration" model of literary composition (reproduced from Hayes [99]).	15
2.2	Motive-level "topic-elaboration" analysis of Mozart melody.	17
2.3	Phrase-level "topic-elaboration" analysis of Mozart melody.	18
2.4	A first-order Markov model of the opening phrase from Mozart's 40th symphony.	26
2.5	An n -gram model capable of reproducing the Mozart phrase with higher probability (i.e., than a 1 st -order model).	27
2.6	Tree diagram of a sentence produced by the generative grammar given in Tables 2.2 and 2.3 (reproduced from Nierhaus [177]).	31
2.7	Output from EMI demonstrating various techniques, including "unifications," "earmarks," and "signatures" (reprinted from Cope [51]).	42
3.1	An example of "virtual polyphony" in the Preludio from Bach's E Major Partita for violin, BWV 1006.	59
3.2	Phrase structure in Bach and IDyOM.	66
4.1	An overview of the MusiCog architecture.	73
4.2	An example of the Beat Entry-Delay music descriptor.	76
4.3	The <i>Beat ED</i> retains the relationship between rhythm and beat, whereas the IOI does not.	77
4.4	The <i>bED</i> also retains correct beat relationships in syncopated contexts.	78
4.5	The rhythmic expectancy function, based on Desain's "basic expectancy" from his "(De)composable Theory of Rhythm."	83
4.6	The decay of an element's cognitive salience in WM as a function of its initial salience ϵ^{sal} , the duration for which it has been retained in WM τ , and habituation.	92

4.7	Chunking in the WM is based on the recognition of segments in the LTM.	95
4.8	Higher-level chunks are formed by non-contiguous repetitions.	96
4.9	Parallelism in Mozart's 40th Symphony. The asterisk indicates the point at which the musical parallelism will indicate a phrase boundary.	97
4.10	Encoding phrases as a combination of Motifs (L_1) and Boundaries (L_2).	100
4.11	L_1 of a CM trained on the opening of Mozart's 40th Symphony, showing all three tiers.	101
4.12	Automatic classification of segments at tier 1. Nodes $\eta_3^1(8)$ and $\eta_3^1(0)$ act as terminals for pairs of closely related segments.	102
4.13	Different approaches to encoding form in the CbCM and the CM.	103
4.14	The second, novel melody can be inferred by the same L_2 schema structure as the Mozart melody.	103
4.15	<i>Schema</i> view of a trained CM showing <i>transitions</i> (grey arrows), <i>terminal links</i> (dotted) and <i>cueing links</i> (black arrows).	104
4.16	Detail view showing the <i>cueing</i> of sequence ((0 - 0 +) (+ - -)).	105
4.17	A hypothetical CM graph in which the out-edge weights indicate a high-probability transition to an L_2 boundary node, thus terminating the L_1 segment.	113
4.18	Selection of an L_{n-1} segment from L_n boundary C . The algorithm must account for L_{n-1} segment probabilities, terminal probabilities, and connectivity to H via cueing links.	115
4.19	Choosing CM transitions with support from edges, terminal links, and cueing links.	117
4.20	The melodic fragment represented in CM graph a expresses a clear parallelism via the sharing of terminal X with boundaries T and U . However, the probability of generating of this parallelism is jeopardized by subsequent learning of melody b	118
4.21	Adapting to unexpected PE segmentation of feedback from the PM.	121
5.1	PE voice-separation of Bach's BWV 70 chorale.	125
5.2	PE voice-separation of complex polyphonic material from Bach's BWV 846 Fugue.	126
5.3	Low-level boundary detection for Bach's BWV 846 Fugue (monophonic arrangement).	128
5.4	Low-level boundary detection for the main theme from Mozart's 40th Symphony.	128

5.5	Low-level boundary detection of Maxwell's <i>Invidere</i> , for flute solo.	129
5.6	A test of the PE's mode and tonal centre induction function.	130
5.7	Mode and tonal centre induction of the opening melody from Mozart's 40th Symphony.	131
5.8	Evolution of the cognitive salience over time for the opening segment ("Seg 1") in Mozart's 40th Symphony. The rate of decay is slowed in m. 7 by the imitative restatement of segment 1, transposed down one scale-step.	132
5.9	Evolution of WM capacity during training.	133
5.10	Formation of chunk structure in WM after the first training iteration on the melody from Mozart's 40 th Symphony.	134
5.11	Formation of chunk structure in WM after training on Mozart's 40 th symphony is complete.	135
5.12	Low-level segmentation of the theme from Bach's BWV 846 Fugue showing the state of hierarchical learning after the first training pass.	136
5.13	Low-level segmentation of the Bach theme after training is completed.	136
5.14	Development of the pitch and rhythm models when trained consecutively on the four movements of Bach's A Minor Partita, BWV 1013.	137
5.15	Nodes added at the Schema, Invariance, and Identity tiers (pitch model) when trained on Bach's A Minor Sonatas and Partita, BWV 1013.	138
5.16	Learning in the pitch and rhythm models when trained on the Finnish folk song corpus ($\varphi = 0$).	140
5.17	The distribution of nodes learned at levels 1, 2, and 3 of both the pitch and rhythm models, when trained on a corpus of jazz songs.	141
5.18	By altering low-level segment structure, higher φ values can negatively impact the detection of higher-level parallelisms.	141
5.19	Excerpt of a "random" generated melody.	144
5.20	Matrix of similarity ratings by Gonzalez Thomas et al.	145
5.21	Comparison of entropy and expectancy-based complexity ratings for MusiCog's generated folk melodies, a folk melody training corpus, and a set of "random" melodies.	146
5.22	Multi-dimension scaling plot of the Folk corpus, the MusiCog generations, and the set of random melodies.	148

5.23	Repeated generation (1000 melodies) with feedback learning enabled produces a clear and significant increase in the complexity and entropy ratings of the generated melodies.	149
5.24	Folk song at median complexity from training corpus.	150
5.25	MusiCog folk generation number 43.	150
5.26	The manual inclusion of a harmonic outline highlights the implied tonal structure of the MusiCog melody.	151
5.27	The Krumhansl-Kessler key profile for C major.	152
5.28	Histograms of the strength of implied tonality (using the <code>maxkkcc</code> function) of the Folk corpus, the MusiCog generations, and a set of random melodies in C major.	153
5.29	Analysis of the segment structure of MusiCog generation number 43.	155
5.30	Plot of complexity and entropy for a corpus of 22 jazz songs and 22 32-measure ‘songs’ generated by MusiCog.	158
5.31	MusiCog’s 18 th generation when trained on the Jazz corpus.	159
5.32	Multi-dimensional scaling plot of the 4 movements from Bach’s BWV 1013 flute partita, MusiCog’s generations, and 4 random melodies.	160
5.33	An excerpt of MusiCog’s 3 rd generation when trained on Bach’s BWV 1013.	161
5.34	Plot of the complexity and entropy ratings for Maxwell’s <i>Invidere</i> , MusiCog’s generations, and 10 random melodies.	163
5.35	Score excerpt from MusiCog’s 10 th generation when trained on Maxwell’s <i>Invidere</i> , for flute solo.	164
5.36	Score excerpt from Maxwell’s <i>Invidere</i> , for flute solo.	164
6.1	Complexity and entropy for all corpora and all MusiCog generations.	166
6.2	Arbitrary rhythmic recombination of isochronous rhythmic segments produces a syncopated overall phrase structure.	168
6.3	Result of segment recombination from an isochronous musical source.	169
6.4	“Virtual syncopation” in the Allemande from Bach’s A Minor Partita.	170
6.5	An example of MusiCog generating novel intervals through recombination.	173
6.6	Generation of novel IOIs in MusiCog.	174
7.1	Metric Markers in ManuScore.	187
7.2	The “cut-away” score style supports the notion of musical <i>objects</i>	188
7.3	MusiCog’s interpretation of a Gesture Line.	189

7.4 MusiCog's presentation of a set of possible continuations from a given musical context.	191
7.5 MusiCog's representation of a real time continuation in ManuScore.	191
7.6 Assigning Instruments in ManuScore.	192
7.7 Note-attached staccato, accent, down-bow, and tremolo articulations.	192
7.8 Using a <i>Link</i> to apply the pitch contour from one staff to another.	194
7.9 The opening phrase in ManuScore (bottom) and its transcription into standard music notation.	200
7.10 The music at measure 12 in ManuScore (bottom), transcribed as a metric modulation in standard music notation.	202
7.11 Basic motives exploited by MusiCog's autonomous generation process for <i>factura iii</i>	203

List of Algorithms

4.1	Rhythmic quantization	79
4.2	Calculate melodic voice-leading cost and cohesion	82
4.3	Update cognitive salience of elements in WM	91
4.4	Update contents of stream	94
4.5	Parallelism-based chunking of elements in a stream	98
4.6	The main WM update function	106
4.7	Learning and inference of L_1 segments	107
4.8	Learning and inference of higher-level segments	109
4.9	Predictive generation from a given state	112
4.10	Selection of L_{n-1} path given L_n boundary	114
4.11	Generate path to a given terminal	115

Preface

I am a composer by training and practice and, like many composers of my generation, I owe a great deal of my early compositional development to two very simple technologies: pencil and paper. As a teenager, spurred by an obsessive love for Mozart's 40th symphony, I carried with me for several months a deteriorating, spiral-bound Passantino manuscript notebook, naïvely scribbling down my own G Minor symphony, in stern defiance of my tenuous-at-best understanding of what "composing," "G Minor," and "symphony" really meant. Countless trips to the public library and a growing collection of music on vinyl, several more Passantino notebooks, pencils, and an untold number of erasers (Stravinsky is reported to have said that music is composed "avec la gomme") were all witness to this early exploration. But another presence loomed on the cultural horizon. Like all "Gen X-ers," I had the curious pleasure of living at the dawn of the so-called "Information Age": an era of rapid technological and cultural change, spawned by the birth of the personal computer. For me, the technologization of music had a humble beginning. A Yamaha QX-1 MIDI sequencer and a Roland JV-1080 sound module became my personal orchestra, as I transcribed music from my Passantino books into the two-line LED display of the QX-1, eager to glean some sense of how these works sounded. From that moment I was trapped.

Twenty years later, the computer had become deeply entwined with my compositional thinking. Though my process was far simpler and more conventional than those you might call "algorithmic" composers, certain aspects of my musical language were now so intricately bound to the computer and its software, that extricating myself from the relationship seemed impossible. It was at this point—in this state of capture—that my current investigation began. What I envisioned when I set out on this journey was a software program that could serve as a kind of compositional "collaborator"; a system capable of contributing in a meaningful way to my creative process. Viewed from the outside, it seemed that generative music had come far enough that a viable method for creating such an "agent" must surely exist. But as my search deepened, it became clear that this was not necessarily the case.

After all, when choosing a collaborator—a human collaborator, that is—it seems natural to demand a degree of competence as a prerequisite. But, if competence for a composer is the ability to compose, then existing generative music approaches clearly lacked musical competence. It was at this point that I stumbled across a book by David Cope called “Computer Models of Musical Creativity” in a London bookshop. Here was a description of musical thinking that was so clear to me, so transparently aligned with my own experience of the compositional process, that a solution seemed imminent.

However, I soon realized that Cope’s approach raised more questions than it could answer. The underlying intuition seemed sound, but the methods employed concealed such a vast wealth of Cope’s personal musical knowledge that their application seemed intractable. And so my research shifted from a focus on how intelligent software tools might function as compositional collaborators, to a much deeper investigation of how musical competence might be implemented in a computer system. This new path of research began with a focus on musical hierarchy and the representation of high-level musical form, but over time a much larger field of exploration was exposed. This dissertation tracks the development of my thinking through a number of important stages in this exploration, which brought me into contact with music psychology, theories of the creative process in music and literature, machine learning, cognitive modelling, and algorithmic composition. The result is a significant first step toward a better understanding of how music perception and cognition might provide a path toward machine creativity in music, opening up the potential for more meaningful human-machine collaboration.

Chapter 1

Introduction

In 1805 Celmenti and Co. and T. Lindsay of London published a volume with the unlikely and impossibly verbose title:

The Melographicon: A new musical work, by which an interminable number of melodies may be produced, and young people who have a taste for poetry enabled to set their verses to music for the voice and piano-forte, without the necessity of a scientific knowledge of the art. [40]

While the notion may seem preposterous to some, the fascination with creating formal systems for music composition is by no means new. Not long before the Melographicon, “Musical Dice Games”—music composition systems moderated by chance, first attributed to Johann Philipp Kirnberger [177]—became popular pastimes at bourgeois parties. A similar impulse some 200 years earlier led Athanasius Kircher to design his “Arca Musarithmica”; a system of wooden sticks (syntagmas) with various markings, which could be used to combine pitch and rhythm patterns with liturgical text for the creation of counterpoint in the styles of the *contrapunctus simplex* and *floridus* [177]. Likewise, around 650 years before Kircher, Guido d’Arrezzo developed a system for deriving melodies given the syllabic content of religious texts using look-up tables and a precompiled set of pitches and melodic phrases [177]. Leaping forward approximately 1000 years, composer, theorist, and programmer David Cope unveiled his “Experiments in Musical Intelligence” (EMI) [48]; a computer program capable of generating complete works in the style of famous composers from the Western classical tradition. Cope was also the first to put a name to the underlying 1000-year-old technique, aptly coining the term “music recombination” [48]. The compositional power of EMI was a breakthrough in its time, and remains essentially unchallenged in

the quality of its musical imitations to this day. However, the basic technique, which involves the fragmentation and recombination of existing music, is fundamentally the same as that employed by the anonymously authored Melographicon, the various dice games, Kircher's Arca Musarithmica, and d'Arezzo's look-up tables. The common underlying intuition that unites all of these systems is their unanimous recognition of the combinatorial nature of music. Though it may be empirically comprised of individual sounds organized in time, the vast majority of music we hear and appreciate as listeners is fundamentally composed of musical *concepts*; mental objects, or "gestalts," with specific attributes and identities that can be apprehended, categorized, and recognized by the human mind. Further, all of the generative systems mentioned above, from the tables of d'Arezzo to the source code of EMI, also have one significant attribute in common; their reliance on the expert knowledge of their authors. As such, there is an ontological gap in their derivation, which cannot be bridged by analysis of their formalisms alone. Even EMI, implemented in a medium as concrete and logical as computer code, cannot be completely understood through objective analysis, since the organization of its algorithms and the selection of musical materials for its underlying database depend entirely on Cope's expert musical knowledge and intuitions as a practicing composer.

In the mid-1800s, at the dawn of the age of mechanical computation signalled by Sir Charles Babbage's "Analytical Engine" (and quite possibly inspired by the potential of the Melographicon), Ada Augusta, mistress of Babbage and the first programmer of the engine, expressed her musical aspirations for the machine as follows:

The operating mechanism [of the analytical engine] [...] might act upon things besides number, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations and which should also be susceptible to the action of the operation notation and mechanisms of the engine. Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent. [86]

Although Augusta never realized her vision of machine composition, an important distinction can be made between this approach and the compositional systems that preceded it. Augusta's notion relies not on the recombination of human-composed musical fragments, but rather on an abstraction of the "fundamental relations of pitched sounds" such

that they may be made “susceptible to the action of the operation notation and mechanisms” of the analytical engine. Though a functioning system was never produced, it is nevertheless informative to consider Augusta’s conceptualization of the problem. First, implicit to her formulation is the question of the *music representation*; i.e., how musical information must be encoded in order to be computable by (or “made susceptible to”) the engine. This aspect of formalization is only indirectly addressed by the earlier systems, which utilize music notation¹—to be interpreted by a human user—as the representational format. Complementing the music representation side of Augusta’s conception is the problem of determining the “action of [...] mechanisms” required for composition; i.e., how formal processes might operate on the music representation in order to produce new music. In this way, Augusta’s conception captures the more general idea of what we now refer to as *algorithmic* music—of music as a “science of operations”—which was to become the mainstay of musical formalization some 200 years later, with the advent of the digital computer. Of course, we cannot know what role expert knowledge might have played in such a system, but it is clear that the level of abstraction proposed exceeds previous approaches. As a curious end note, Augusta’s use of the phrase “scientific pieces of music” is also intriguing, as it suggests a role for computation in the composition of new forms of music, unrestrained by previous notions of musicality, and accountable only to the “fundamental relations of pitched sounds in the science of harmony.” This notion of using computation for the invention of entirely novel musical forms was to enter musical discourse only some 100 years later, through the work of composers like Iannis Xenakis and theorists like Otto Laske, who sought to use numeric formalization as an aide for exploring new musical territory.

1.1 Reformulating the Problem: A Holistic Approach

Acknowledging that modelling music composition in a formal system is a long-standing challenge, the following dissertation takes a step back to look at the problem from a holistic perspective. Specifically, we take an integrated cognitive modelling approach (see Section 3.7), grounded in experimental evidence from the music perception and cognition literature, as a strategy for building a formal system that acknowledges the psychological underpinnings of human listening. The central contribution of this work is an integrated cognitive architecture for music called “MusiCog” which, to the best of our knowledge, is the first

¹Kircher’s system does contain a degree of numeric abstraction, in that it uses scale-steps to represent pitches and symbolic music notation values to represent rhythmic quantities.

of its kind. The rationale for this approach derives, in part, from an understanding of the active role—attested to in the field of music psychology—that listening plays in the comprehension of music. The complexity of the listening process requires a form of cognitive organization which, we suggest, makes the fundamental knowledge required for compositional thinking accessible even to untrained listeners. This is not to deny the high level of skill, knowledge, and training required of expert composers, but simply to highlight the degree to which such specialized knowledge draws on the fundamental perceptual and cognitive capabilities demonstrated in music listening. Through the integrated approach we are able to examine the way in which largely innate perceptual and cognitive processes inform the structure of musical knowledge and in turn support basic forms of compositional thinking. Thus, composition in MusiCog does not claim to model the thinking of an educated composer, versed in music theory and composition, but rather models the thinking of a “musically naïve” composer, informed purely by listening, and the implicit musical knowledge that can be gained therefrom.

Chapter 1 begins our investigation by considering the idea of composition in its broadest sense, with a focus on common conceptions from the fields of music and literature. We look at several approaches to music generation, and consider them from the perspective of their composition theoretical underpinnings. While we acknowledge that such approaches do not necessarily propose to model composition directly, we suggest that the composition theoretical perspective provides a useful analytical tool for examining the role reserved for compositional thinking in system design. To conclude this chapter, we give a brief overview of three established theories of creativity, providing a context for understanding the affordances given by a formal system for producing creative compositional behaviour.

In Chapter 2 we outline several key ideas from the field of music psychology that have directly informed our work on MusiCog. This section provides a brief introduction to relevant research in music perception and cognition while also providing essential background information for understanding MusiCog’s design. We discuss three existing, cognitively-grounded systems for music generation, and give a brief introduction to the field of Integrated Cognitive Architectures (ICA), which directly influenced the design of MusiCog. Chapter 3 provides a detailed technical description of MusiCog, beginning with the music descriptors used to represent input. We then provide descriptions of each of the main processing modules, and conclude with a brief discussion of relevant implementation details. In Chapter 4 we offer a number of test cases examining the perceptual and cognitive processing of musical input provided by each of MusiCog’s modules, including

the learning of hierarchical musical structure, and the generation of melodic output. These early compositional efforts of MusiCog are reviewed in the context of *musical style imitation*; that is, we evaluate MusiCog's ability to compose novel melodies in the style of melodies that it has "heard" previously. In Chapter 5 we discuss the strengths of the integrated approach, and provide an overview of questions and challenges raised by this work. The chapter closes with a brief discussion of MusiCog's capacity for creative behaviour. The training files, generated output examples, score excerpts, and software downloads referenced in this dissertation can be found at the following link:

<http://www.sfu.ca/~jbmaxwel/MusiCog/index.html>

In Chapter 6 we introduce a Computer-Assisted Composition (CAC) system called *ManuScore*, which utilizes MusiCog as a generative "agent." ManuScore's main features are outlined and two musical works composed in ManuScore are discussed. We also provide results from a listener study conducted in a live concert setting, in which music composed using ManuScore was presented in public performance. Our discussion of ManuScore concludes with a few observations regarding the score representation used, and how this appeared to promote a compositional practice focused on music perception rather than on the theoretical structures implicit in common music notation. Chapter 7 concludes the dissertation with a discussion of future work and a few closing remarks on the current state of MusiCog's development.

Chapter 2

Composers and Authors Compose Compositions

The verb “compose,” with its c.1400 roots in the Old French “composer,” means to “put together, arrange, write” [1]. The word combines the Latin *com-* “with” and *+ponere* “to put, place” (as in “position”), and is commonly associated with the term “composite,” meaning “to put together, to collect a whole from several parts” [1]. The musical sense of the term, with its (English) agent noun form “composer,” dates back to 1590, and from 1640 on it saw more general application as “one who combines into a whole.” The noun form, “composition,” dating from the late 14th century, carries the rather reflexive definition of “a thing composed of various elements,” but also inherits earlier 13th century associations with “agreement” and “settlement.” The still older Latin root *compositio* incorporates the notion of “connecting.” Usage from the 1550s suggests an expressly literary meaning, designating composition as the “art of constructing sentences,” and around 1600 the notion of “literary production”—in some cases specified as “a school exercise in the form of a brief essay” [240]—was used. The usage associated with the visual arts, “arrangement of parts in a picture” [1], appeared later, in 1706. Composing can thus be broadly defined as the act of collecting, connecting, placing, constructing, combining, and putting together several parts, through agreement or settlement, into a whole. It may be noted, however, that although contemporary definitions of composition clearly acknowledge the notion of *creation*, this idea is conspicuously absent from the term’s etymological roots. The suggestion is that compositions are not strictly “brought into being,” as created things must by definition be, but are instead comprised of existing parts or elements, which in some sense precede the compositions themselves. Although composition is certainly understood to be a creative

act, and the composer must ideally bring together elements in some artful way, the definition implies that the elements themselves enjoy a somewhat independent existence; they are singular mental concepts, and composition is fundamentally recombinant.

Although common usage of the term “composer” refers predominantly to musical practice, reference to both literary and musical *products* is made with the term “composition.” The same is true of the verb form “compose”—one composes a song, letter, symphony, or poem. Of course, it is also common to speak of “writing” music. But “to write” is by definition associated not so much with literary composition, *per se*, but with the physical need to “carve, scratch, or cut” [1] symbolic characters into some fixed medium when recording ideas or recounting events for posterity. When referring to strictly literary composition, a somewhat more grandiose designation is made, by use of the term “author.” Considered from its etymological roots, an author is literally the “founder, enlarger, master, leader,” and the “one who causes to grow” [1]. The Old French “autor” carries a slightly different connotation, referring to the “originator, creator, instigator” [1]. It was only later, in the 14th century, that a specifically literary usage appeared as “one who sets forth written statements” [1]. Just as the term “composer” became associated with musical composition, it is clear that the term “author” has become commonplace when referring to literary composition. Yet few would argue that composers are commonly authors of musical works, and authors composers of language. So what sort of tacit understanding led to the rather conspicuous delineation of these two terms, composer and author, along musical and literary lines, in spite of the fact that both disciplines are understood to produce “compositions?”

In making his case for a genuinely compositional perspective on music theory, composer and theorist Otto Laske noted that:

Strangely enough, composers usually accept the notion that listening, not composition, is the paradigmatic musical activity on which a theory of music is to be based. [135]

However unpalatable assertions like “composers usually accept” may be, Laske nevertheless makes an interesting point about the apparent indivisibility of composition and listening in music theoretical discourse of the late twentieth century [135]. Indeed, contemporary definitions of music, like Edgar Varese’s famous notion of “organized sound,” generally accord a central role to audition in characterizing music. More recently, neural imaging studies [103] have revealed that imagined music also relies heavily on regions of auditory cortex, suggesting that even in the absence of sound music remains inherently concerned

with audition. This shouldn't be particularly surprising since it's difficult to imagine how a mental representation of music might come into being, let alone be communicated to another, without some encounter with auditory perception. Yet, it certainly hasn't always been the case that music and audition were so inextricably linked. Indeed, the Pythagorean definition of music suggests a much grander notion of universal order, quite removed from contemporary notions of music as "organized sound." The Pythagorean philosopher and early music theorist Boethius conceived of three types of music: *musica mundana*, *musica humana*, and *musica instrumentalis*. *Musica mundana*, the so-called "music of the spheres," described the harmony of the celestial order, while *musica humana* concerned the harmony of the human body and the relationship between body and soul. Only *musica instrumentalis* corresponded to what we define as music today; it was the only "musica" that one could hear. However, this should not suggest that audibility was of marginal importance to Boethius and the Pythagoreans. On the contrary, it was the role of *musica instrumentalis* to make the true mathematical order of the universe (*musica mundana*) accessible to the senses [101]. Thus for Boethius and the Pythagoreans, music was primarily a field of mathematics investigating the laws of harmony, which, though made observable through sounding bodies, was not fundamentally restricted to the domain of sound.

Music psychologist Diana Deutsch notes that this Pythagorean notion of music as mathematics has never ceased to guide the practice of music theory:

The view that music ought to be investigated solely by contemplation of numerical relationships has characterized most music theory since Pythagorean times. [...] Also stemming from the mathematical approach of the Pythagoreans have been the various attempts to build entire musical systems by mathematical deduction from a minimum number of established musical facts. [63]

As a scientist, Deutsch expresses a certain discomfort with this state of affairs, suggesting that such theories should be treated as hypotheses, subject to verification through scientific experimentation [63]. Indeed, such an approach might have led to a very different development of musical language from that witnessed during the 20th century, with its progressivist focus on mathematical formalization and systematization. Reflecting this mathematical inclination of music theory, there is clearly an extra-auditory focus to systems like serialism, or the Schillinger system [207], which provide compositional and organizational constraints that operate quite independently of audition. These Pythagorean aspects of music theory would thus seem to contradict Laske's complaints about a disproportionate

emphasis on listening in theoretical discourse. Indeed, much recent work in generative music is likewise focused on developing parsimonious, mathematical formalisms, which avoid heuristics based on musical or compositional knowledge. While this tendency can be attributed, in part, to a general widening of the research field, drawing in greater numbers of non-musicians—computer scientists, neuroscientists, engineers, etc.—it also reflects an underlying cultural fascination with the Pythagorean notion of music. Echoing back to Boethius, such purely mathematical models relegate listening to a passive, receptive role, in which the listener bears witness to the higher mathematical order of the universe, as evidenced through musical sound.

But it is important to recognize that listening is not, from a psychological perspective, a purely passive experience. Peterson presents a compelling theory of the inherent creativity of music listening, based on the notion that listeners actively create mental models of music as it unfolds [188]. During this process, listeners select salient “musical objects” (i.e., via attention), relate them to one another, categorize them, and transform them in various ways, according to their individual musical knowledge and experiences. Peterson suggests that the process of developing a mental model of a given musical work is inherently creative, and follows a pattern similar to the compositional process of many composers, starting with a general, schematic overview of the work, and proceeding to unfold the moment-to-moment details through repeated exposure. Further, it has been shown that even passive listening activates premotor areas of the brain [17], particularly in cases where somewhat predictable rhythmic patterns are involved, further suggesting a latent form of “action” inherent in music perception. Of course, such knowledge of the neurophysiology of auditory imagery, or the presence of premotor activity during listening is relatively recent, and was not available to Laske when he first expressed his opinion that composition should be placed on a equal footing to listening in music theory. And it would be easy to overemphasize Laske’s complaint about the theoretical assertion of listening as the “paradigmatic musical activity.” If his position seems reactionary from a contemporary perspective, it nevertheless represents an understandable response to the dominant view of his time, given the limited body of research on listening processes available to him (and to music theory in general). In fact, as his thoughts about “composition theory” matured, it became clear that Laske saw any either/or theoretical bias toward listening/composition as arbitrarily limiting: “neither the paradigm of Composition nor Listening provides insight into the whole of music; each yields a pair of glasses of limited vision” [137].

2.1 Laske's Composition Theory

Laske saw composers as experts in “virtual” or “possible” music, which he felt required an alternative approach to a tradition based on the exploration of existing music—a practice he felt could only serve to limit musical creativity [135]. Laske associated “virtual music” with the idea of “inner hearing,” which included, but was not limited to, imagining¹. Though, as Zatorre and others have shown [103, 251], imagining music has much in common with hearing music, from a neural perspective, so that the two ought not to be placed in mutually exclusive categories. Another important component of Laske's definition of virtual music, as it relates to composition theory, arose through the integration of the computer as a compositional tool. Following Chomsky, Laske characterized the cognitive demands of composing in terms of three primary factors: *competence*, *performance*, and the *task environment*. Competence was defined by the relevant knowledge and skills of the composer, performance by the application of such knowledge and skills during composition, and the task environment by the set of tools and materials used for composition. Laske felt that the type of abstract thinking required for the specification of musical knowledge in algorithmic processes—and composition software in particular—was a defining characteristic of contemporary practice. Such thinking focused on reasoning about the task environment and the selection of materials and procedures best suited to realizing a particular musical goal.

Whether used in sound or score synthesis, and in whatever mode of interaction, programs have [...] forced musicians to focus on the pro-active, rather than the re-active, aspect of their activity, and have given them a chance to choose, rather than suffer, their processes. [137]

Laske proposed two fundamental modes of compositional thought: “example-based” and “rule-based.” Example-based composition is grounded in knowledge of previous music, which is conceptualized in terms of “sound objects.” Since these objects are recalled without access to the knowledge structures that brought them into being, Laske classifies example-based models as “data models.” Rule-based models of composition, on the other hand, are considered “process models,” because they characterize music according to the sequence of operations, or “decision rules” that inform its development. In this sense, rule-based models represent music using “procedural objects” [137]. Laske does not, however,

¹Laske initially treated these terms synonymously, but gradually came to differentiate them, so that the computational representation of music was considered “virtual music,” and imagined music was considered “possible music.”

assume that procedural models alone embody a complete knowledge of music, but rather that they presuppose data models, since the procedures involved always have music as their object. Thus the important difference between the data model and the procedural model, with regard to example-based and rule-based composition respectively, lay in the fact that the “steps or rules of the procedural model emphasize, not the structure of the data model, but its use in (real) time” [137].

In formulating a practice of rule-based composition, Laske places a strong emphasis on the notion of design. According to its dictionary definition, to design is to “decide upon the look and functioning of [an object], typically by making a detailed drawing of it” [65], and has its root in the Latin *designare*, meaning “to mark out.” The term thus carries a strong visual connotation, paired with an implicit strategy of abstraction, facilitating the evaluation of products before they are committed to their final material form. However, the colloquial understanding of design has broadened considerably over time, moving well beyond its diagrammatic roots. Ralph [195] offers a more explicit, contemporary definition: “a specification of an object, manifested by an agent, intended to accomplish goals, in a particular environment, using a set of primitive components, satisfying a set of requirements, subject to constraints.” Though Laske clearly subscribes to this more inclusive definition, he also partitions his conception of design into two categories: a dualistic conception, and a holistic one. The dualistic conception places a human designer in opposition to an external task environment. The holistic conception, on the other hand, proposes a dialogical relationship between designer and design; a process Laske formalizes through his notion of the *compositional life cycle*: “the design creates the designer as much as the designer creates the design.” In order to understand Laske’s notion of the compositional life cycle, it is important to realize that Laske is speaking with reference to composer/designers, who create their own systems, based on their own set of specifications and requirements. Composers like Hiller, Xenakis, and Cage, through to Ames, Koenig, and Truax all fit in this general category, whether or not their systems were implemented as computer programs. In principle, all such approaches are algorithmic, and embody a similar type of design-based musical thinking. Such composers realized their compositional strategies through abstract modelling of their musical goals, and the resulting designs in turn influenced the trajectory of their compositional development. The dialogical aspect of this process is what Laske termed the compositional life cycle: “This metaphor emphasizes the shifting perspective of a designer who finds his competence and innervations reflected in his task environment—such as a program for computer-assisted composition—rather than in examples of existing

music” [137]. Of course, it is worth remembering that a piano, pencil, and manuscript are as much a “task environment” as a computer and its software. But it is generally difficult, if not impossible, for composers using such tools to directly influence their design and structure, or to extend their functionality beyond the intended purpose².

Of particular relevance to Laske was a compositional process in which a software system translated a musical design goal into an abstract, numeric representation. This representation was then translated by the composer, through a process of *interpretation*, into the musical specifics of pitch, rhythm, dynamics, and so on. It is this notion of interpretation that Laske held in the highest regard, suggesting that “expert designers” excelled in their capacity to synthesize complex, and often multidimensional design data, into coherent musical interpretations. This is precisely the kind of process used by composers like Hiller and Xenakis in their earliest experiments with statistical models, and later extended into the domain of CAC software with the work of Truax, Koenig, and many others since. It is worth noting that Laske’s commitment to representing musical ideas through mathematical abstractions reveals, once again, the Pythagorean roots of his musical (and music theoretical) thinking. In a sense, his conception of holistic design can be said to advocate a balance between *musica mundana* and *musica instrumentalis*. It is his particular notion of the compositional life cycle that actualizes this balance, by cycling between abstract specification and design (*musica mundana*), and the verification of the interpreted musical design data through listening (*musica instrumentalis*). In this light, it could be said to fall to music psychology, and cognitive modelling in particular, to close this Pythagorean musical circle with a model of *musica humana*, bringing the mathematical and the phenomenological—structure and process—together through perception and cognition.

But perhaps the most important aspect of Laske’s proposal was the emphasis it placed on extending the reach of music theory by accommodating the composer’s unique experience of music “as it is being composed” [135]. Laske’s conception of composition theory sought an engagement with the thought processes that inform compositional development; broadly speaking, he promoted a theory of how composers *think music*. In this sense, Laske was advocating a genuinely “authorial” function for music, and promoting an understanding of how the composer is an *originator*, *creator*, and *instigator* of music, rather than one who simply puts together sounds—whether remembered or perceived—to make a whole. His conception doesn’t eliminate listening entirely, but it does propose to supplant

²Composers like Harry Partch, who invented instruments to his musical specifications, which in turn influenced the development of his compositional language, are notable exceptions.

the structural understanding of music that arises through perception (and cognition), via listening, with the highly deliberative notion of composition as design. This is the Pythagorean root of Laske's conception of music. Listening is chronologically secondary, representing the process by which music's higher mathematical order is made available to perception. Composition and listening, for Laske, are thus fundamentally divided.

2.2 Composition Theory in Literature

Contemporary literary composition theory predates Laske's formulation for music by approximately 10 years, having its inception in Flower and Hayes' seminal 1981 work on *cognitive process theory* [85]. This work sought to interrogate the dominant model of creative thinking of the time, broadly referred to as "stage theory" [236]. Stage theories suggest that creative thinking proceeds through a series of behaviours, each of which corresponds to a stage of completion in the growth of a creative product. The series of stages can be recursive, potentially returning back to earlier phases of behaviour during the completion of the creative task, but it is nevertheless implicitly linear. Flower and Hayes' model, on the other hand, proposes that "writing is best understood as a set of distinctive thinking processes which writers orchestrate or organize during the act of composing" [85]. These thinking processes are embedded hierarchically within one another so that, for example, generating ideas may be embedded in a planning process, or grammatical construction embedded in a "translating" process. Such hierarchies, however, are not assigned an *a priori* order of precedence, and a given process may be embedded within any other process. The main high-level processes identified by Flower and Hayes are *planning*, *translating* (i.e., of ideas to text), and *reviewing*, all of which are supervised by a separate process referred to as "the Monitor." Such processes have natural subprocesses (though the hierarchies can change), so that planning often involves generating, organizing, and goal setting, while reviewing involves evaluating and revising. The Monitor, as the name suggests, monitors the current process as well as the overall progress of the composition, in order to strategically determine when to change processes. The translating process involves the conversion of ideas into a readable form; a complex task, drawing on knowledge of grammar, semantics, and syntax, and also on motor skills like handwriting or typing.

The task environment in Flower and Hayes' model is represented by the written (or in-progress) text and the "rhetorical problem." The rhetorical problem includes the topic of the writing, the intended audience, and an exigency, expressed in terms of the role of the writer

and the demands of the written work; e.g., the requirements placed on both writer and text are markedly different for a biology student and a journalist. Competence is represented by the long-term memory of the writer and includes both topical knowledge and writing skill, and performance is represented by the processes outlined above. This kind of formal modelling has allowed theorists to form hypotheses and to evaluate differences between expert and novice writers. A common research strategy involves monitoring writers of different skill levels during writing tasks using a “speak aloud” protocol, in which participants are asked to describe their thought process as they write. Using this methodology, McCutchen [168] observed that advanced writers tended to produce larger ratios of spoken text to written text, suggesting greater conceptual complexity in formulating the rhetorical problem, and a more highly optimized realization of that complexity in the written text. Flower and Hayes noted that poor writers tended to rely on abstract high-level goals, like “appeal to a broad range of intellect,” in spite of the fact that such goals did not provide sufficient structure to guide the writing process. Experienced writers, on the other hand, tended to create more constrained, operational goals, like “give a brief history of my job.”

Research has also investigated so-called “capacity theories” of writing, which propose that short-term and working memory capacity directly influence writing quality [223, 26]. Such studies have shown that competing tasks that constrain cognitive resources—written transcription, for example—have a negative impact on quality. This is particularly notable in a developmental context, where it has been observed that as children acquire greater fluency in transcription (i.e., physical writing) the complexity and quality of their writing improves. Swanson and Berninger’s work [223] suggested that this effect was the result of an acquired fluency in transcription freeing up cognitive resources for composition. Similarly, Bourdin and Fayol [26] conducted a serial recall memory task in which children and adult subjects were asked to recall word sequences, both orally and in writing. Though adults performed similarly in the memory task, in both the written and oral conditions, children performed significantly worse in the written condition, suggesting that lack of fluency in transcription was directing cognitive resources away from the memory task. Furthermore, in a second test, which required the adult subjects to write their responses only in upper-case cursive letters—a generally unfamiliar manner of transcription—adults performed at a level similar to children.

In later revisions of Flower and Hayes’ original theory, Hayes [99] restructured the model to operate on three different “levels”; a *control* level, involving motivation, goal-setting, planning, and revision; a *process* level, comprising the original model’s writing processes and

task environment; and a *resource* level, incorporating working memory, long-term memory, and other aspects of competence. At the process level, in addition to the original notion of “translating,” Hayes included a category for transcription, which emphasized the mechanical processes of text generation (including spelling and orthography, but also writing technologies) and the cognitive resources such skills require (as illuminated by the experimental work outlined above). Of particular relevance to our discussion of algorithmic music, Hayes also developed a basic computational model of child-level literary composition. His goal was to investigate aspects of Bereiter and Scardamalia’s “knowledge-telling” model; a text generation process in which “spreading activation” in long-term memory causes a chain of associations that help keep the written text on topic without prior specification of a high-level plan (i.e., as required by Flower and Hayes’ model). Hayes, however, recognized that a common pattern in children’s writing was unaccounted for by knowledge-telling. In this pattern, first identified by Fuller [88], the main topic is elaborated through a knowledge-telling process, which in turn introduces and elaborates a series of additional subtopics, after which the writer returns to the main topic. Hayes defined this type of text as a “topic-elaboration text,” which he implemented in a computational model, along with two simpler models capable of producing two additional types of texts; “flexible-focus” texts and “fixed-topic” texts.

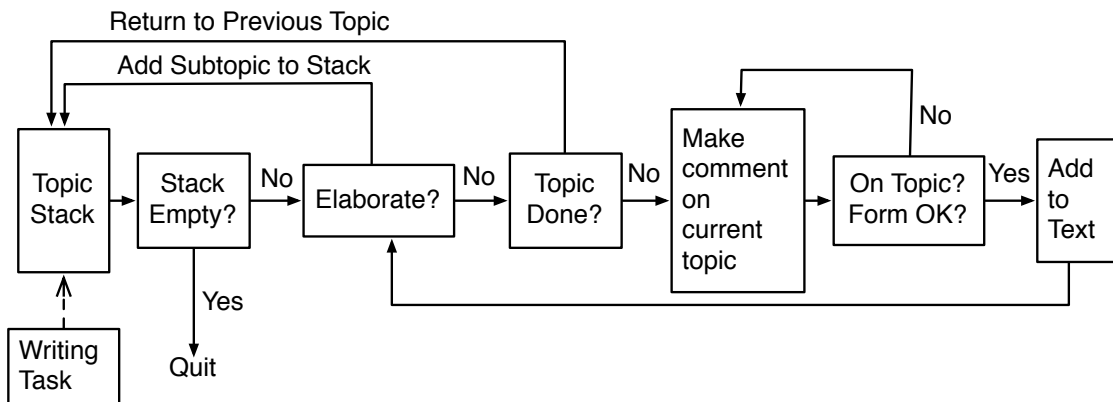


Figure 2.1: Hayes’ “topic-elaboration” model of literary composition (reproduced from Hayes [99]).

The flexible-focus model replicates the development of an initial topic through chains of association, as in the knowledge-telling model. The model begins by making a statement about an initial topic that is used to recall associated information from long-term memory.

It then produces a statement about the retrieved information, creating a new “focus” for the text, and continues in this manner until the text is deemed complete. The fixed-topic model, on the other hand, only generates statements about the initial topic. Although this process may seem more simplistic, in fact it is more complex, since it introduces an evaluation process, used to determine whether recalled information is relevant to the original topic. The topic-elaboration model, shown in Figure 2.1 (reproduced from [99]), is the most complex, introducing an additional “topic stack,” representing a form of working memory and serving as a planning structure. Topics can be added to the topic stack through “elaboration,” and can be removed from the stack when knowledge of the topic is exhausted (i.e., using the “topic done” function). This model is capable of producing nested structures of topics and subtopics, the complexity of which is a direct function of working memory capacity and the amount of topical knowledge stored in long-term memory. Hayes notes that modelling younger writers requires a smaller capacity for the topic stack, while older writers can be modelled using a larger stack [99]. The topic-elaboration model was used to generate the text in Table 2.1, using a simple long-term memory comprised of a predefined set of topical statements. In this text, the sequence of topics—Frank (1), computer (2), Christmas (3), computer (2), Frank (1), dog (4), Frank (1)—reveals a nesting process requiring the functioning of working memory in both the writer and the reader for comprehension. Together, the three models implemented by Hayes produced text structures consistent with the analysis of 96% of Fuller’s corpus of children’s texts [99].

I like Frank because	1
Frank got a new computer.	2
It was a Christmas gift.	3
Christmas is my favourite holiday.	3
I hope we will have a white Christmas.	3
His computer is a laptop.	2
Frank has a dog.	1
The dog has four puppies.	4
Frank is my best friend.	1

Table 2.1: A text with topics and nested subtopics produced by Hayes’ “topic-elaboration” model. The column on the right indicates the index of each topic in the set of topics used.

Hayes’ topic-elaboration model is of particular relevance to music composition, which often exhibits a similar formal development, both at the phrase level, and at hierarchically

higher levels of structure. For example, if we substitute the notion of “topic” for that of the musical “motive,” we can observe topic-elaboration processes at work in the opening theme from Mozart’s 40th symphony, as shown in Figure 2.2³. Here we see a simple melodic motive introduced, elaborated, and then restated a few measures later. A similar pattern can also be seen at a higher formal level, if we enlarge the structure of the “topic” to include the entire opening phrase, as shown in Figure 2.3. It is important to note that such a formal correspondence does not assume a semantic (or even syntactic) correspondence between language and music. It simply suggests that both mediums share similar conceptions of form, and that this similarity can likely be attributed to the resource dependent cognitive constraints at work in both composer/author and listener/reader.

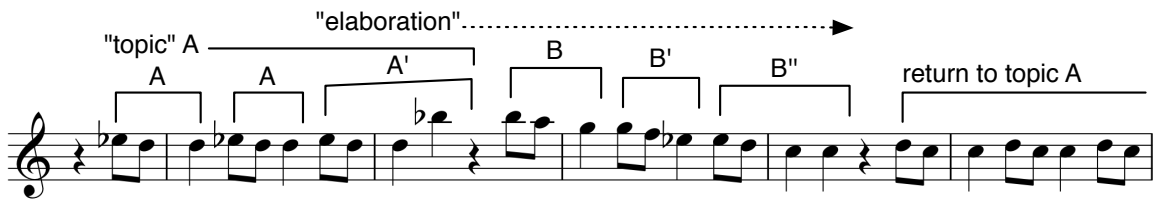


Figure 2.2: Motive-level “topic-elaboration” analysis of Mozart melody.

Somewhat curiously, Laske never mentions literary composition theory, though the emphasis on mental processes in this field clearly has much in common with Laske’s thoughts about rule-based composition and procedural models. Similar to Laske, such models focus less on the structure of the data model and more on the rules and procedures used to organize that data when generating texts. Literary composition theory also acknowledges the constraints placed on the writing process by working memory capacity, particularly with regard to the cognitive resources required for translating and transcribing. This is a category of “competence” that Laske does not directly address. On the other hand, because Laske makes a special affordance for the role of the computer in musical composition theory, at least some of these constraints can presumably be addressed computationally, shifting their influence from the domain of competence to that of the task environment.

Perhaps the most compelling aspect of literary composition theory—and Flower and Hayes’ cognitive process model in particular—is the way in which it connects the mental processes of reader and writer through a shared reliance on basic cognitive resources.

³It is worth noting that music is capable of quite subtle formal elaborations, since pitch and rhythm patterns need not be varied simultaneously. For example, in the Mozart melody, the phrase labelled “B” utilizes the same rhythm as phrase “A,” simultaneously suggesting both repetition and change.

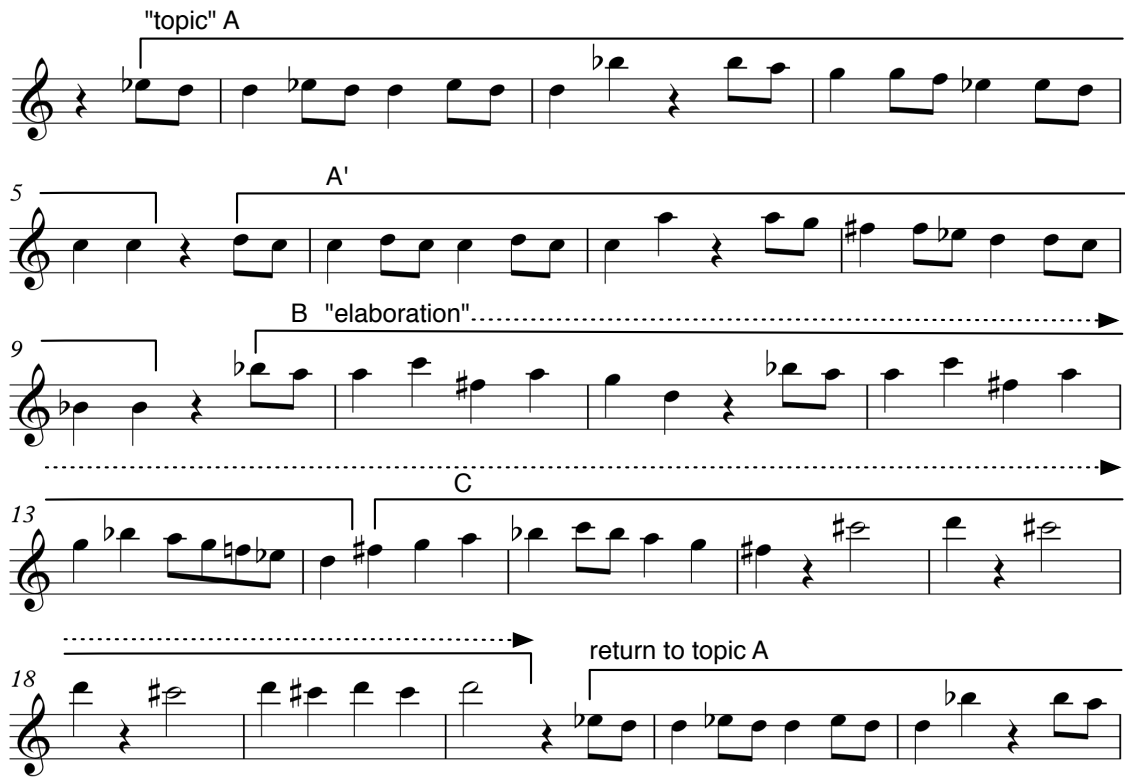


Figure 2.3: Phrase-level “topic-elaboration” analysis of Mozart melody.

Composition arises from the world model stored in long-term memory, but is dependent upon working memory for its expression in the form of the written text. Comprehension for the reader is likewise dependent upon cognitive resources, and in a manner that directly reflects the dependencies of the writer, as encoded in the semantic and syntactic structure of the text. The relationship between writer and reader thus forms a symbiosis, grounded in architectural aspects of cognition; that is, on the interdependence of long-term memory (topical knowledge), perception (seeing/hearing the written text), working memory (retaining and relating salient information), and action (writing and/or reading).

2.3 Collins’ Synthesis Process Model

In his work on musical creativity, David Collins [41] proposes a model of the creative process in music composition drawing on elements of four prominent theories: Wallas’ *Stage*

Theory [236] (mentioned above), Wertheimer's conception of *Gestalt Theory* [75, 241], Gruber's *Emerging Systems Theory* [96], and Newell and Simon's notion of *Information Processing Theory* [176, 213]. An important aspect of the gestalt theory is its conceptualization of creativity as a process of gathering individual elements together into an integrated whole, or gestalt. At a certain point in this gathering process a so-called "flash of illumination" often occurs (also a feature of Wallas' theory), leading to a *restructuring* of the parameters of the problem, and in some cases a complete reconceptualization of the gestalt itself. It is this restructuring of understanding in support of the gestalt that drives creativity, and guides the search for novel solutions to familiar problems. Emerging Systems Theory takes a somewhat different approach, suggesting that creativity is a slow, developmental process, in which many solutions are tried, a select few of which will actually succeed. This trial-and-error process leads to a form of optimization, reminiscent of Darwinian notions of evolutionary development through natural selection and competition. In contrast to the above models, Newell and Simon consider the problem from the perspective of computer simulation, and suggest that creativity can be approached as a special form of problem solving. The information processing approach conceptualizes creativity as a trial-and-error search of a given "problem space," through which the system seeks to arrive at a predetermined "goal state." The search is formalized as a set of serial operations, following a particular rule system, which Collins suggests (following Baroni [14]), in the case of music composition, implies the presence of an underlying generative grammar (i.e., in order to define the legality of movements in the problem space). The ill-defined nature of the goal state in many creative tasks—particularly those involving artistic creation—is addressed primarily through the notion of novelty or newness; i.e., a novel solution is a valuable solution. Collins suggests that it is only through a synthesis of the above theories that we can adequately explain the kinds of processes observed during human music composition.

Collins also draws attention to the methodological difficulties involved in data collection from human compositional processes, and emphasizes his motivation to extend previous approaches, both in terms of the detail of the data collected, and the naturalness of the compositional task performed. Through an intensive, three-year study with a professional composer, he tracked the composition of complete musical works, collecting data through three techniques: 1) The use of MIDI "Save As" files, 2) verbal protocols, and 3) interviews and verification sessions. The MIDI data files were used to capture the state of the developing composition incrementally throughout the compositional process, allowing Collins to analyze the chronological development of the musical work, and to observe any significant

changes of structure that might occur. Verbal protocols are a common data collection technique in the social sciences, often involving self-reports of the subjects engaged in carrying out assigned tasks (as in Flower and Hayes' work in literary composition [85]). In this case, Collins used "immediate retrospective reporting," in which the subject gives a report on the work done immediately following the work session (i.e., not during the working process, as in the "speak-aloud" protocol used by Flower and Hayes). The interviews and verification sessions were primarily directed toward verifying the verbal reports, and also to verifying the results of a "structural mapping" process that Collins carried out, using both the MIDI data files and the verbal protocols. The mapping process focused on three types of correlation: 1) "Real-Time Mapping," which mapped verbal reports to the content of the MIDI data files; 2) "Thematic Mapping," which mapped frequently-occurring keywords in the verbal reports (e.g., "evolve," "add," "mood," etc.) to comments of the composer regarding his compositional processes and strategies—i.e., in terms of the arrangement of materials, selection of dynamics, instrumentation, pitch content, and so on; and 3) "Structural Mapping," which concerned the overall structural development of the work, in terms of the relationship between low-level changes to musical details and high-level restructuring. The composer was allowed to compose freely, with no limitations on his working hours, or on the instrumentation or duration of the work, which was to be used as a "showreel" for a computer game soundtrack.

An interesting finding of Collins' analysis was his observation of the considerable creative effort and activity that arose as a direct result of the composer's reluctance to alter or abandon two main themes that had been composed early in the process, on the first day of work. This hesitation led to considerable thematic restructuring and alteration of the composer's process, suggesting that these two "seemingly incompatible" themes were somehow essential to the composer's overall conception of the piece. Reorganization of the themes **A** and **B**—specifically, by reversing their order of appearance in the work (**B**, **A**)—also led to a subsequent "problem proliferation," in which low-level details of each theme's setting had to be altered in order for the two themes to function in the restructured high-level form. This restructuring also led to the "inspired" decision to introduce an altered version of theme **A** at the beginning of the work—resulting in an (**A'**, **B**, **A**) form—providing a smoother introduction, while also musically "hinting" at what was to come. Collins refers to this as a process of "reformulating the givens," in which prior assumptions about the nature of the materials, or the overall form, are altered or abandoned, in order to promote the

generation of novel and/or better formed solutions. Several such stages of problem proliferation and reformulation are outlined in Collins' analysis, revealing a complex, cyclical process in which novel solutions lead to new problems, which in turn demand the formulation of further novel solutions.

Thus, although Collins' study echoes the work of Flower and Hayes in its characterization of the creative process as a set of hierarchically related processes and subprocesses, the complexity and detail of Collins' observations suggests a more fluid, adaptable process. In Collins' model, the creative process follows a cyclical, iterative pattern involving four main component processes:

1. Determination of a set of "germinal" themes, ideas, and/or motifs
2. Determination of a set of goals and/or sub-goals (e.g., "extend section," "improve closure of theme," "adjust harmony")
3. "Problem proliferation," caused by the need to reconcile different technical requirements intrinsic to the ideas and goals emerging from stages 1) and 2)
4. Definition of a "solution space" in which problems introduced in 3) are addressed, generally leading to a re-structuring of the goals and sub-goals of the composition

As can be imagined, the result of stage 4) has the potential to lead back into stage 2), which in turn can lead to a dramatic expansion in the complexity of the compositional task in stage 3). Thus, the compositional process generally cycles through successive phases of reformulation, often (but not always) involving moments of gestalt understanding or "illumination." Such moments may alter the composer's overall conception of how the various themes of the work function, and how they contribute to the whole, and can therefore lead to dramatic restructuring of previously composed materials. Another important aspect of Collins' model is the implicit emphasis it places on recording elements of the work in progress (i.e., of "transcription" in Hayes' model). Capturing the raw materials of the work during its development allows the composer to evaluate decisions in a more objective manner—i.e., from a listener's perspective—without needing to retain the details in memory, thus freeing cognitive resources for problem reformulation and the search for novel solutions. Finally, there is also a strong acknowledgement of the iterative nature of the compositional process, and the non-linearity of musical decision making.

2.4 Composition Theory and Generative Models

Whether a system is explicitly designed to model compositional thinking or not, it is nevertheless the case that any system for the automatic generation of music is *implicitly* a theory of musical composition⁴. In positing an organizational structure for music, and formalizing a process for algorithmic generation from that structure, systems designers must necessarily make decisions, and with each decision they reveal the fundamental philosophy of musical creativity that lies beneath. To decide—from the Latin *decidere*, literally “to cut off”—is to wield Occam’s famous “razor,” and to accept that a multitude of alternative theories, explanations, or models must thereby be ignored. Thus, whether the proponents of predictive statistical models of generative music, for example, genuinely advocate the notion that prediction is a fundamental compositional strategy, they do support the principle of parsimony that underlies the statistical approach [217, 237]. Since statistical analyses of musical works provide compelling descriptive models of the structure of music (see Temperley [226] and Huron [108] for a full discussion), it seems natural to suggest that they might also offer productive prescriptive models; i.e., plausible accounts of how music might be composed. If we are to explore the notion that generative models embody theories of composition, it will be useful to examine a few popular approaches to music generation, and to consider how those approaches conceptualize the process of composing music.

2.4.1 Composition by Prediction: The Markov Property

By far the most thoroughly explored models for music generation are those classified as Markov models. Named after Andrey Markov, who famously created a statistical model of the letters in Alexander Pushkin’s *Eugene Onegin*, Markov models came into prominence with the publication of Claude Shannon’s seminal work on “information theory” [212]. A Markov model is any system in which the probability of the future state of some random variable X can be predicted by the variable’s current state; i.e., $P(X_{t+1} = S_j | X_t = S_{i,t})$. Accordingly, the “Markov property” is attributed to any example of X for which its future state depends only on its current state. Markov models are implemented using a finite set of symbols called an “alphabet” Σ , representing the possible states of random variable X , and a transition table S_{ij} for recording the frequencies of state transitions. In a typical implementation, X is a single symbol from Σ , representing the most recent state. Such

⁴This is the case even when a system is devised for generating a single composition since the formalism used expresses the design considerations implicit in the compositional idea.

a model is referred to as a “1st-order” Markov model, since it uses only one state for the prediction of the future state (a “zeroth-order” model, on the other hand, is equivalent to the histogram of X ; i.e., the time-independent frequency of occurrence of all states of X). In an n^{th} -order Markov model, rather than considering only the state at time t , a “memory” of n previous states (X_{t-n}, \dots, X_t) is created. By representing a larger history of X , and thereby creating a more discriminating context for X_{t+1} , higher-order Markov models gain predictive power. Because such models use a finite history of the music being modelled to predict the future state, Conklin refers to them all under a general class of “context models” [44].

Another approach to building higher-order models is through the use of “ n -grams.” In n -gram models, X is a string, representing a concatenation of symbols of length $0 \leq n \leq N$, where n is the “order” of the model and N is a maximum size parameter (usually only 3 or 4 symbols in musical models). N -gram models are common in natural language processing, where the lexicon of the language being modelled provides an *a priori* set of viable n -grams. For example, in English, the transition (w, h) represents a unique phoneme, and is thus guaranteed to occur with sufficient regularity to justify the definition of “wh” as an n -gram; e.g., (wh, o) , (wh, y) , etc. The n -gram is defined as a state of variable X , and the resulting probability distribution treated as 1st-order Markov model; i.e., S_{ij} will record transitions from the given n -gram to a subsequent symbol. The order of a model can also be adapted dynamically, in order to maintain predictive power without arbitrarily increasing the size of Σ . In so-called “variable-order Markov models” (VOMM), the order is increased only in cases where predictive power will be gained. For example, a 2nd-order model, which can reasonably differentiate between “who” and “why,” loses predictive power for words like “when” and “wheel.” In such cases, the probability of “when,” for example, must be calculated as the joint probability $P(e|wh) \times P(n|e)$. The loss of predictive power (specifically, an increase in entropy, or uncertainty) arises from the fact that, in standard English, $P(X|wh)$ is part of a distribution represented by words like “when,” “which,” “who,” “what,” and so on, and the same is true for $P(X|e)$. By increasing the order of the model, and creating the n -gram “whe,” the entropy associated with $P(X|wh)$ and $P(X|e)$ is removed, leaving the much lower entropy space represented by $P(X|whe)$. In the English lexicon, this represents a considerable improvement in predictive power over the 2nd-order version, while increasing the size of Σ by only one n -gram.

In generative music algorithms, the Markov model has been a standard tool since the early experiments of Hiller and Baker. For their 1963 *Computer Cantata*, for example, they used a statistical analysis of Ives’ *Three Places in New England* to build zeroth-, 1st-, and

2nd-order Markov models. Using these models, they extracted selection probabilities to be imposed over pitch, rhythm, duration, note versus rest, and playing style [6]. Markov models have also been incorporated into commercially available generative music systems, like Zicarelli's *M* and *Jam Factory* [253]. Typically, such applications use the predictive capacity of the Markov model to stochastically generate a musical continuation (e.g., a new MIDI note) given the current musical context. This approach can be related to the notion of "melodic expectancy" in the music psychology literature, commonly associated with Narmour's Implication-Realization model. Early experimental work, most notably by Krumhansl [124], provided compelling evidence in support of Narmour's principles, using "probe tones" to evaluate listener preferences for different continuations of a given musical context. In such experiments, listeners were played a short musical fragment followed by a single probe tone, and were asked to evaluate the fitness of the tone given the musical fragment. It was found that listeners tended to favour those tones that followed Narmour's principles of melodic expectancy. However, later work investigating the statistical properties underlying such judgements showed that a statistical learning mechanism could often provide a better explanation of this behaviour than Narmour's principles [186, 202]. If we assume that musical well-formedness is a correlate of listener expectancy, then such findings appear to support the idea that statistical prediction could provide an appropriate strategy for musical composition. That is, if listeners evaluate the well-formedness of musical passages by statistical inference, then stochastic selection of continuations from a statistical model ought to generate expected musical results. Indeed, it was this kind of intuition that led to the strong interest in Markov models in the years following the publication of Shannon's theory [6].

However, it is important to clarify the distinctions between the compositional application of Markov processes and the ideas underlying Shannon's original theory. Shannon's theory was not focused on prediction, *per se*, but rather on data compression and/or the successful transmission of information over noisy channels. For compression, the theory sought to quantify the limits within which a data compression algorithm could accurately (in the case of "lossless" compression), or with an acceptable degree of error (in the case of "lossy" compression), represent a message using the minimum number of bits. In the case of data transmission, the objective was to enable the extraction of an original message (or some intelligible approximation thereof) in the presence of noise. Thus, from a psychological/cognitive perspective, Shannon's objective is perhaps better understood as recognition or interpretation than prediction. That is, given a noisy representation of the

symbol at time t , we must be able to make a reasonable guess as to its identity (i.e., using available information at time $t - 1$). Or, in the case of compression, an algorithm must not compress beyond the point where the original message can still be retrieved from its compressed representation (again, often allowing for some acceptable degree of error). The important point is that, in both compression and transmission, some form of the original message is used as a control (i.e., to assist with error correction). This general application is also echoed in the experimental use of probe tone judgements mentioned above, where the tone is offered as a continuation of a given musical context; by analogy, it is a more or less “noisy” transmission of a musical “message.” In such a situation the listener evaluates the well-formedness of continuations in a manner not unlike Shannon’s statistical receiver. However, is it necessarily the case that, given the option to select a continuation independently, listeners would express the same underlying statistical regularities?⁵ And further, if they did, would this be equivalent to composition?

Of course, the reconstruction of a musical message is explicitly *not* the objective in Markov-based music generation, where it is assumed that there is no “original” message; i.e., the generated message must itself be original. In this context, it is generally felt that the guide for successful “retrieval” is simply a notion of acceptable aesthetic quality (which is poorly understood), and that this should be used as a criterion for determining the order of the model (which has a strong influence on the model’s compression rate⁶). For example, it is generally accepted that 1st-order models compress too aggressively, and generate arbitrary or chaotic musical patterns, while higher-ordered models generally reproduce the original message too faithfully, leading to frequent musical “quotations,” and consequently a loss of musical originality.

As an example of the difference between these notions of message interpretation and prediction, consider a model trained on the prior example of the opening theme from Mozart’s 40th symphony. The 1st-order model can be represented as in Figure 2.4. If we consider a situation in which some form of distortion is imposed on the musical signal—for example, performance by a student violinist, prone to fingering and intonation errors—we

⁵It is worth noting that this kind of “compositional” approach to measuring expectancy has been conducted [33, 210], involving musical continuations either sung or performed on the piano. Although similar results have been obtained, however, this approach poses methodological problems due to its dependence on performance expertise, altered attention (i.e., switching from listening to composing), and physiological constraints (e.g., individual vocal range).

⁶For example, a 1st-order model can encode the pitch information of all known equal-tempered musical compositions using only 88 symbols—i.e., the notes on a conventional piano. However, such aggressive compression would introduce a high degree of entropy, resulting in a model of limited usefulness for musical generation.

can demonstrate the use of the Markov model to correctly interpret the musical message. Say, for example, that the first note is played correctly, but the second note is slightly sharp, producing a pitch lying between $E\flat$ and D. Applying the Markov model, we see that the probability of transitioning from $E\flat$ to $E\flat$ is zero, whereas the probability of transitioning from $E\flat$ to D is 100%. Thus the model can be used to verify the intended pitch, effectively “correcting” the error, and restoring the integrity of the original musical message. If, on the other hand, we use the model to generate a 10-note melody and apply the product rule to calculate the probability of generating the original, we find there is only a 0.2% chance⁷ that we will correctly reproduce it. Of course, as mentioned earlier, the goal of Markov composition is not to reproduce an original melody. But such an example nevertheless demonstrates an underlying problem with prediction as a compositional strategy; the statistical analysis offers a quantifiably valid *descriptive* model, but does not provide a sufficient *prescriptive* model. That is, the model can explain what happened in the melody, but it does not provide sufficient information to reproduce it with acceptable probability. Considering this problem from the perspective of Hayes’ topic-elaboration model for literary composition, it could be said that the difficulty lies in the fact that a simple Markov model offers no conceptual category analogous to a musical “topic,” and no decision process for connecting topics, if a topic were to be found. It can be argued that a higher-order model, like the one in Figure 2.5, would improve the chances of reproducing the original. However, since the fundamental problem of the predictive approach remains unaddressed (i.e., descriptive adequacy and prescriptive inadequacy), the same problem will surface again, at higher levels of melodic form.

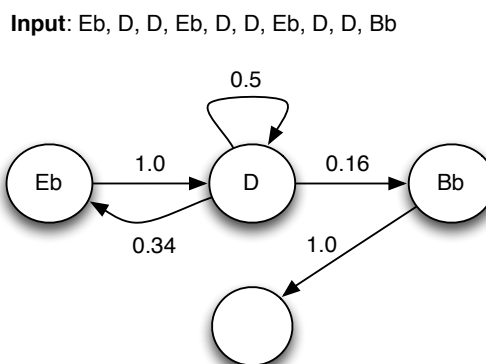


Figure 2.4: A first-order Markov model of the opening phrase from Mozart’s 40th symphony.

⁷ $1 \times 0.5 \times 0.34 \times 1 \times 0.5 \times 0.34 \times 1 \times 0.5 \times 0.16 \times 1 = 0.0023$

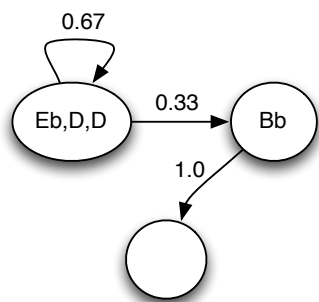


Figure 2.5: An n -gram model capable of reproducing the Mozart phrase with higher probability (i.e., than a 1st-order model).

More complex Markov models, like Hidden Markov Models (HMM), or Conklin and Witten’s “Multiple-Viewpoint” context models [45], can improve the situation somewhat by accounting for factors that, while not represented directly by random variable X , nevertheless have an influence on the state of the variable. In the case of the HMM, a series of observations is paired with an underlying, but not directly observable, Markov model that is hypothesized to have a direct influence on the sequence of observations. It is then possible to use the joint probability of the observed and hidden states to infer a specific hidden state as the cause of an observation. Allan [3] used an HMM to harmonize chorale melodies in the style of J.S. Bach by treating the notes in the melody as the observed states, and a 1st-order Markov model induced from the analysis of Bach chorales to represent the sequence of hidden states. This is an appropriate use of an HMM since one of the model’s primary functions is to calculate the most probable sequence of hidden states, given a sequence of observations [194]. Of course, such a process is not essentially predictive since the final state of the inferred sequence is the state at time t (i.e., not time $t + 1$). However, prediction is possible, since the hidden states are generally represented using a 1st-order Markov model, and can therefore be used to calculate the probability of state X at time $t + 1$ (i.e., given the inferred state at time t), from which a probable observation can be predicted.

Somewhat like the HMM, Conklin and Witten’s multiple-viewpoint approach also exploits a joint probability, except that in this case the probability is calculated across a set of “viewpoints,” each representing a different musical attribute or “type”—pitch, rhythm, harmony, etc.—of a given event. A viewpoint is a combination of a type, and the context model for that type, induced from analysis of a given corpus. Some types represent the transition probabilities of individual attributes (pitch, for example), while others model interactions between attributes (e.g., pitch and duration) by calculating joint probabilities for two or more

types. These compound types are referred to as “product types,” and are treated as individual viewpoints in the model. During prediction, a weighted linear combination of all viewpoints is used to determine the probability of a given event. There are many ways to determine the weightings used when combining viewpoints. In one approach, the relative entropy for each viewpoint, given a particular context-event pair (c, e) , is used to weight the linear combination of viewpoints, such that higher-entropy viewpoints are assigned a lower weighting. The multiple-viewpoint approach is one of the few techniques that can account for the dynamic interaction between various musical attributes of pitch, interval, rhythm, duration, and so on, in musical structure.

Turning now to the larger subject of composition theory, it is worth noting that predictive methods of generation from statistical models have little to say about compositional process, as it is conceptualized by Laske and Hayes (see Section 2.2). To highlight prediction as a fundamental strategy is to suggest that compositional thinking is primarily automatic and reactive. Considered from the perspective of Hayes’ models of literary composition, the predictive approach suggests that music composition follows a “flexible-topic” model; a musical idea is introduced, stimulating associations in memory, from which a highly-activated association is selected and appended to the composition, and so on. Though there is perhaps a degree of truth in this, particularly during divergent, exploratory work involving improvisation (i.e., where learned skills of instrumental performance have a strong influence on output), it provides little support for the deliberative problem solving aspects of composition highlighted by cognitive process theories. From a psychological point of view, the predictive approach suggests that musical knowledge is largely implicit and procedural, with little recourse to conscious musical decision making. The popularity of this approach is likely due to the drive for parsimoniousness that motivates much of the computational modelling of music, along with the relative ease of implementing fast and efficient stochastic learning and generation methods using Markov models (facilitating their use in “online,” interactive systems). Statistical approaches are also seen as plausible descriptive models of cognition, owing to support from a large body of experimental evidence highlighting the psychological capacity of human listeners to infer statistical regularities over a broad range of musical attributes—melodic, harmonic, and rhythmic—and across a wide range of musical cultures (for an overview, see [219]).

It is important to note, however, that the evidence pointing toward a statistical mechanism underlying human musical expectations does not in itself justify the singular application of prediction as a compositional strategy. Conklin notes that the common technique

of random-walk generation, which epitomizes the predictive approach, generally fails to produce “high-probability music,” in large part because it relies upon a greedy algorithm [43]—i.e., choices are made based on local probabilities only, without heuristics for optimizing the probability of the complete generated sequence. It could be argued that a similar shortcoming explains the insufficiency of prediction as a theory of human composition. Alternative stochastic techniques can avoid the problems associated with random-walk methods, but these are not fundamentally predictive. In time, it may be revealed that music, rather than representing the high-probability “emissions” of some beautiful and elegant predictive system, may instead exist as an elegant representation of a fundamentally chaotic, fickle, messy, and frequently inelegant human process.

2.4.2 An Innate Faculty for Music: Generative Grammars

A class of generative models for music that is formally related to, but philosophically distinct from, the various Markov models are those systems known as generative grammars. The formal specification of grammars is usually attributed to linguist and philosopher Noam Chomsky, who proposed the model as a way of explaining the human capacity to generate novel syntactical structures, given only limited examples of syntactically correct natural language statements [38]. This capacity was attributed to an innate language faculty, “hard-wired” into the human brain, representing a set of basic rules for manipulating symbols. This idea stands in opposition to purely statistical accounts of the brain’s language capacity, under the argument that children, for example, can create novel natural language statements using structures that they could not have learned through statistical inference alone, having not been exposed to a sufficient set of example statements.

Formally, grammars are expressed through a set of “rewriting rules,” which allow symbols in the given language to be rewritten with new symbols or groups of symbols. Chomsky identified four types of grammars, based on the structure of the rewriting rules: Type 0 “unrestricted” grammars, Type 1 “context-sensitive” grammars, Type 2 “context-free” grammars, and Type 3 “regular” grammars. The current discussion will look only at Type 1 and 2 grammars, as these have been applied directly to musical problems. The formatting of the rewriting rules define the syntax of the language. This formalism allows for the implementation of two primary functions: 1) the generation of novel statements in the given language and 2) the verification of arbitrary symbol sequences as syntactically valid (or invalid) statements in the given language. The rewriting rules follow the general pattern that symbols on

the left-hand side of the rule expression can be rewritten using symbols on the right-hand side. As an example (from Nierhaus [177]), take the rule-set in Table 2.2.

S	→	NP	VP		
VP	→	V	(NP)	(PP)	
AP	→	(Adv)	A	(PP)	
PP	→	P	NP		
NP	→	(DET)	(AP)	N	(PP)

Table 2.2: A set of rewriting rules describing the “non-terminal” symbols in a generative grammar (from Nierhaus [177]).

where S = sentence, NP = noun phrase, VP = verb phrase, V = verb, PP = prepositional phrase, Adv = adverb, A = adjective, P = preposition, AP = adjectival phrase, DET = article (“determiner”), and N = noun (bracketed symbols are optional). The items in this rule-set (specifically, parts of speech) are referred to as “non-terminal” symbols, since they will not appear in any final production of the grammar. The various parts of speech can be rewritten according to the rule-set in Table 2.3, which defines the set of “terminal” symbols for the grammar. Given a set of terminal and non-terminal symbols, and their associated rewriting rules, syntactically valid statements can be generated, following a process similar to the one illustrated in Figure 2.6.

N	→	man	girl	John
DET	→	a	the	
V	→	met	saw	
A	→	nice	good	quick
Adv	→	very	extremely	
P	→	in	for	to

Table 2.3: A set of “terminal” symbols to be used with the generative grammar in Table 2.2 (from Nierhaus [177]).

In this type of grammar, construction of syntactically valid statements proceeds in a top-down fashion, starting with the definition of a structure to be generated (sentence “S”), and progressing through a series of substitutions that gradually fill in the specifics (i.e., terminals) of the generated statement. Grammars of this type, where the rules are pre-defined, are referred to as knowledge-based [177]. It is also possible to derive rules from a

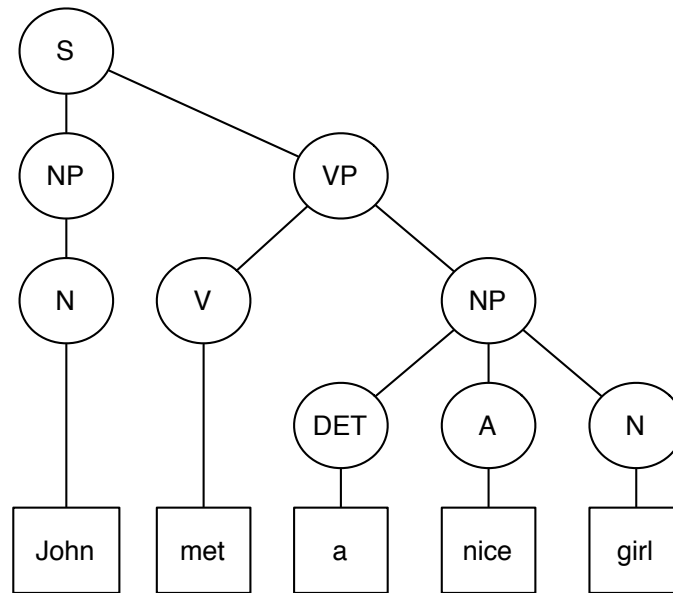


Figure 2.6: Tree diagram of a sentence produced by the generative grammar given in Tables 2.2 and 2.3 (reproduced from Nierhaus [177]).

corpus of examples through grammatical inference. Kohonen proposed a context-sensitive musical grammar, the rules of which could be learned from a given corpus [119] using a technique referred to as the “dynamically expanding context” method. This method maps the shortest unique context to an immediately subsequent production. This approach is, of course, fundamentally the same as the method used when building Variable-Order Markov models, as outlined in Section 2.4.1; i.e., varying the order of the n -gram at time t in order to reduce (or remove) the entropy of the distribution at time $t + 1$. There is an important difference, however, between Kohonen’s grammar and the one demonstrated in Figure 2.6; whereas Kohonen’s grammar is “context-sensitive” (Type 1), the grammar from Nierhaus’ example is context-free (Type 2). This is to say that productions from Kohonen’s grammar are, strictly speaking, continuations of a given musical context and, as such, depend upon the context for their syntactic validity. Further, the context in Kohonen’s grammar is always retained in the final musical statement; i.e., both the context and the production are terminal symbols. This absence of non-terminal symbols and strong context-sensitivity again reveal a close connection between Kohonen’s model and the class of Markov models. In Nierhaus’ context-free grammar, on the other hand, the productions are instantiations of grammatical functions—e.g., “verb phrase”—represented by non-terminal symbols, which may be given

a broad range of realizations without sacrificing syntactic validity. The non-terminal symbols thus represent a level of abstraction not demonstrated by Kohonen's model.

Kippen and Bel's "Bol Processor" [15] is also a context-sensitive grammar, but unlike Kohonen's grammar, which operates only on terminal symbols, the Bol Processor utilizes a combination of expert knowledge and grammatical inference to define its terminal and non-terminal symbols. The system was designed to model the complex compositional and improvisational practices of North Indian tabla drumming. Because this tradition defines certain classes of patterns and certain methods for transformation, variation, and recombination of those patterns, it lends itself to grammatical approaches, where the relationships between patterns (and their variations) can be represented using non-terminals, and their contents represented using terminals. Due to the high complexity of the patterns used in this music, Kippen and Bel found it untenable to discover and represent a comprehensive rule set via expert knowledge alone. For this reason, they used an interactive, human-moderated machine-learning process, in which experts were asked to evaluate both valid and invalid patterns discovered by the system, which in turn used the resulting expert knowledge for the inductive inference of an underlying formal grammar. Processes for inferring the grammatical structure of music, in consideration of certain aspects of expert knowledge, can also be applied to the practice of Western tonal (or "functional") harmony. In music of this tradition, harmonic structures have specific functions based on their positions in the diatonic scale. Harmonic theory, which developed in dialogue with the compositional practice of composers over a period of several centuries, gradually formalized a set of basic relationships between the chords of different scale steps (or "roots"), such that viable sequences of chords could be derived from abstract formal plans based on root movements alone. Recognizing that such an explicit formalism could be expressed grammatically, Rohrmeier developed a generative grammar for Western harmonic practice [199]. Having its foundations in harmonic theory, this grammar is also knowledge-based, but Rohrmeier's approach is broad enough to be applied to the analysis of a wide range of tonal genres. An important aspect of Rohrmeier's grammar, which differentiates it significantly from Kohonen's, is that it acknowledges the non-sequential, hierarchical aspects of harmonic relationships. As an example, Rohrmeier notes that in the common harmonic sequence (I, II, V), the harmonic function of the II chord may be better understood as a precursor of V than a "production" of I; that is, II is often used to "prepare" V, in spite of having a somewhat ambiguous relationship to I. Indeed, recognizing the potential of the II chord's syntactic double-role, in which it was re-conceptualized as the "V of V" (i.e., since

the II chord lies five scale-steps above the V chord)—composers of the romantic era carried this notion of preparatory functions to its logical limit, allowing them to produce long passages of continuous harmonic development. Given only the sequential dependencies acknowledged by the Markov property this kind of formal extension would have been quite impossible, as such a representation does not offer an equally informative conceptualization of the sequence as that offered by the grammatical model.

It is worth pointing out that Rohrmeier's grammar has similarities to Lerdhal and Jackendoff's well known and highly influential Generative Theory of Tonal Music (GTTM) [146]. Somewhat like Rohrmeier's grammar, the GTTM takes an analytical approach to building hierarchical descriptions of musical form, grounded in principles of music perception and cognition. The model considers four main types of formal structure: "grouping structure," which parses the music into fragments like *motives*, *phrases*, and *sections*; "metrical structure," which highlights the interplay of stressed and unstressed musical events at different hierarchical levels; "time-span reductions," which assign structural importance to pitches according to the grouping structure and metrical structure; and finally "prolongational structure," which orders pitches hierarchically according to notions of "tension," "relaxation," and "duration." In order to account for the strong context-sensitivity of musical form, the GTTM also employs a set of "preference rules," which represent the selection criteria of the listener, used when organizing the above structural influences according to their impact on the perceptual coherence of the music. While the GTTM provides a strong descriptive theory, it lacks the prescriptive power of Rohrmeier's model, since it is not explicitly defined through a set of rewriting rules (as required by a formal grammar).

More recently, Rohrmeier has extended his investigation of musical grammars into the field of music psychology. In a 2011 study [200] Rohrmeier, Rebuschat, and Cross exposed musician and non-musician listeners to a corpus of melodies generated from a synthetic finite-state grammar. The grammar was designed to avoid patterns typical of Western music, with the intention of isolating the influence of learning (rather than prior musical knowledge and experience) on listener evaluations. Both "grammatical" and "ungrammatical" melodies were included in the training set, where grammatical melodies employed the finite-state grammar, and ungrammatical melodies varied from close approximations of the grammar to randomly selected terminals. Participants were divided into experimental and control groups, and the experimental group was given preliminary training via the presentation of 17 grammatical melodies. During training, participants were also assigned a concurrent tone-counting task as a distractor, in order to promote an incidental learning context

(i.e., they could not concentrate exclusively on the content of the melodies). The results showed that listeners were able to differentiate grammatical from ungrammatical melodies, and that this effect was not dependent upon prior musical training. Thus, in cases where the underlying musical system is unfamiliar, musical expertise appears to have little effect on performance. An unexpected but significant effect of online learning was also observed, in which participants in the control group (i.e., untrained) showed a significant improvement in performance over the course of the testing phase, indicating that listeners were able to induce this novel grammar in a relatively rapid manner.

A related study by Loui et al. found similar evidence for the induction of musical grammars in human listeners [152]. In this study, the researchers used a synthetic scale based on a 13-note subdivision of a “tritave” (i.e., a 3:1 ratio version of the Western 2:1 “octave”), upon which they built a harmonic framework of three-pitch structures arranged in low-integer ratio relationships. In this way, the new system could represent harmonic structures analogous to “triads” in Western diatonic music. Two closely-related finite-state grammars were created, based on the assertion that melodic structures in tonal music are monophonic articulations of an underlying harmonic progression. The two grammars were thus differentiated only by the harmonic progressions that each class of example melodies articulated. As in the study by Rohrmeier et al., participants demonstrated the ability to learn the underlying structure of each grammar by successfully differentiating between example melodies from each system. Because the tuning system was foreign to all subjects, effects of long-term enculturation to particular pitch patterns and relationships could be effectively ruled out. Both studies show that higher-level organizational systems for music can be quickly inferred by listeners, and that such systems provide perceptual and cognitive cues for the evaluation of syntactical structure. The study by Loui et al. is particularly compelling, as it suggests that formal systems operating at higher levels of abstraction, like that of Rohrmeier’s context-free grammar of functional harmony [199], are supported by a robust cognitive learning mechanism operating in both trained and untrained listeners.

From a composition theoretical perspective, grammars are interesting because of their capacity to address the non-linear, iterative aspects of compositional thinking observed by Collins (see Section 2.3), while still acknowledging the linear/sequential nature of music. It could thus be argued that grammars model both the abstract, conceptual aspects of compositional thinking, and—particularly in the case of context-sensitive grammars—the sensitivity to local temporal dependencies that constitute the experience of the listener. Thus, the grammar’s rewrite rules can be used to propose an abstract formal solution to

a given musical problem, while the concrete solution can take into account local factors. Whether to accept or reject a musical solution, of course, remains an aesthetic matter, but the grammar provides a degree of confidence in the viability of a given choice. Proposed solutions need not rely exclusively on local sequential dependencies, as in random-walk predictive methods, but can rather be reviewed in full consideration of the outcome of a particular musical decision. Such is the case with Rohrmeier's example of the (I, II, V) progression, where the II provides a "solution" to the "problem" of transitioning from I to V. Since the arrival at V is already known, proposing the II as a solution suggests a hierarchical decision process. It should be noted that, due to its frequency of use in tonal music, the harmonic sequence (I, II, (V)) would also likely be a high-probability choice for a Markov model. Thus, it is the conceptual difference that is of interest here; the grammatical representation implies a hierarchical knowledge of harmonic relations (i.e., the role of the II chord as the "V of V"), which the Markov model cannot capture. It is possible, in principle, for a grammar like Rohrmeier's to produce a secondary dominant of this kind in contexts not represented by a given corpus, and thus inaccessible to a statistically-induced Markov model using random-walk generation.

The non-linear, iterative, and atemporal⁸ aspects of formal grammars also align with aspects of Laske's notion of "rule-based" composition, where the rule set has the potential to represent a space of possibilities not previously conditioned by the constraints of existing music⁹. In Laske's practice, the numeric representations generated by the rules (as embodied by a computer system; see [137]) are analogous to the "non-terminals" of the grammar, and it falls to the human composer to "interpret" these into the sequences of terminals representing concrete musical statements. Through the iterative refinement of the compositional life-cycle, composers gradually define patterns of interpretation, thus actualizing a complete—if intuitive and unstated—grammar, existing somewhere between the composer's competence and the task environment represented by the computer system. The central role of iteration is also acknowledged in process-based models of creativity, like Flower and Hayes' model for literature, or Collins' for music, which invariably highlight

⁸Xenakis referred to these kinds of abstract formal entities—the kind that are representable as non-terminals—as "outside time," a category to which he assigned scales, modes, hierarchical forms, fugal architectures, and so on [249].

⁹It is worth noting that the grammars implied by Laske's conception are actually knowledge-based, but with the rather unusual caveat that the "knowledge" applied is not directly derived from the language being modelled (i.e., traditional conceptions of music), but is rather imported from some other conceptual domain. In this sense, the grammatical aspects of Laske's conception are somewhat metaphorical.

the role of iterative development in the refinement of rhetorical/formal functions and the resolution of syntactic/semantic problems.

2.4.3 Iteration Toward Musical “Fitness:” Evolutionary Models

Inspired by Darwin’s history-changing book, “The Origin of the Species,” the field of Evolutionary Computing (EC) draws on methods analogous to processes of natural selection. There are two primary streams of work in this area: Genetic Algorithms (GA) and Genetic Programming (GP). We’ll look at GA first. The basic GA routine proceeds as follows:

1. Generate an arbitrary population of individuals
2. Using a fitness function, select individuals for mating
3. Generate a new population through sexual reproduction and mutation
4. Repeat from step 2 until some termination condition is reached

This deceptively simple algorithm can provide an extremely powerful and effective heuristic search function¹⁰. Which is to say that, if the evolution of an “individual” capable of representing a particular solution is possible, then a well designed GA will eventually find that individual. In order to better understand why this is so, it’s worth looking more closely at what happens at each step in the basic algorithm given above (for a detailed introduction see Mitchell [170]).

- **Step 1)** Produces a random population of individuals; in computer implementations, these individuals (more specifically, their “chromosomes”) are often specified as binary character strings—e.g., [0 0 0 1 1 0 0 1]. It is important to note that these are not purely “random” strings, in that their length and type (e.g., binary, real-numbered, ASCII character, etc.) are essential factors in their utility as potential solutions¹¹.
- **Step 2)** Uses a “fitness function” to determine which individuals will be paired together for reproduction. Defining the fitness function can pose particular challenges when designing evolutionary models, and often represents the point at which some form of expert knowledge is encoded into the system. In most implementations, the

¹⁰Heuristics are essentially rules of thumb that can be used to reduce the number of dead-ends (i.e., local minima/maxima) reached when looking for a solution in a large search space. They are particularly useful, even essential, for spaces large enough to lead to intractable “brute-force” searches.

¹¹However, a fitness function that checks for basic representational validity would quickly weed-out individuals that violated such basic structural imperatives.

selection step will not choose *only* the fittest individuals, but will also probabilistically select a certain percentage of less fit individuals for reproduction. In the context of heuristic search, this step helps prevent the system from falling into local minima by occasionally forcing the search into less-than-optimal locations in the search space. Individuals that are not selected for reproduction are culled from the population.

- **Step 3)** The genetic operators of “cross-over” and “mutation,” are used to generate new individuals for the population, given the two selected “parents.” Cross-over is directly analogous to the combination of genes from both parents seen in sexual reproduction, whereas mutation models the random alterations in genetic material that arise through physical processes.
- **Step 4)** Checks whether some termination condition has been reached. In the ideal case, this is a matter of determining whether any individual in the population offers a solution to the proposed problem. However, in cases where an ideal solution is not possible (music composition is a good example, since an optimal “solution” does not, strictly speaking, exist), the termination condition could simply be a predefined maximum number of iterations, or generations. If the termination condition is reached, the algorithm terminates, and if not, it returns to step 2) and continues until a solution is found (or the termination condition reached).

A closely related EC technique, known as Genetic Programming (GP), seeks to evolve optimal *programs* using a similar algorithm to that given above; that is, rather than directly evolving a solution, GP evolves a program capable of calculating a solution. Thus, in GP the chromosome encodes a set of simple programs, or operations, rather than a set of attributes. For example, a GP system might work with a set of simple programming operations like “Add,” “Subtract,” “Multiply,” “Divide,” or any arbitrary predefined function “ $f(x)$.” The chromosome would encode combinations of these operations by assigning each operation an integer label—e.g., Add = 1, Subtract = 2, Multiply = 3, and so on—thus replacing the binary string representation outlined above with a string of integers representing the various “programs” to be run. When modelling more complex operations, GP systems will often specify a “Function Set” defining the set of simple mathematical operations, and a “Terminal Set,” consisting of the discrete values used as arguments for the functions (i.e., the constants and variables of the generated program). In this way, chromosomes can represent sets of programs along with their associated constants and variables, allowing for the evolution of highly complex, algorithmic operations. As can be imagined, one of the

great challenges of GP is to ensure that the evolutionary process always generates syntactically valid, compilable programs. As long as this basic requirement is satisfied, however, the evolutionary process itself operates in fundamentally the same manner as in GA.

A relatively early application of EC in music was Horner and Goldberg's system for computer-assisted composition [104]. In this system, EC is used for "thematic bridging"—i.e., the gradual transformation of thematic material over time. Since thematic bridging itself could be seen as a form of "mutation," this is a straightforward and appropriate test bed for EC techniques. Although Horner and Goldberg discuss the use of GA specifically, their approach in many ways resembles GP, with the definition of a set of "operators" representing basic formal transformations of thematic material—e.g., "Add," "Delete," "Mutate," "Exchange," etc. The chromosome encodes a sequence of operators, to be applied in left-to-right order, thus indicating the order in which the set of operations will be applied during transformation of a given theme. As a result, the chromosome itself provides a set of simple instructions for thematic bridging; a technique more similar to GP than standard GA. Since the goal state of thematic bridging is to achieve the desired thematic transformation, the fitness function used in this case simply measures how close a given generation comes to producing the desired transformation. Of course, this implies that the goal state is known beforehand, and that a reasonable measurement of fitness can be derived from comparing the system's output to the target input. Such well defined notions of fitness are not, however, always possible in EC systems for music composition.

Thus it is clear that one of the most difficult tasks in the application of EC to creative tasks is the definition of an appropriate fitness function; i.e., to pinpoint how to differentiate a "fit" solution from an "unfit" one. A common approach to this problem is to use interactive methods, in which the system passes the role of assigning fitness over to the user, avoiding the need to define a concrete, computable notion of fitness. Such systems generally evolve one or more generations, then present the user with an interface for auditioning the results and selecting individuals for reproduction. This is the approach taken in Johanson and Poli's "GP-Music" system [110]. As the name suggests, GP-Music uses GP techniques for the transformation of melodic sequences, with the subjective evaluation of musical fitness provided by the user of the system. One of the difficulties in this type of interactive approach, however, is that it tends to slow down the evolutionary process considerably, as the system must pause for evaluation by the user before evolving new generations. In Waschka's "GenDash" system [239], on the other hand, the fitness function

is random, and user control is afforded¹² only through the selection of the initial population. Similarly, Eigenfeldt's "Kinetic Engine 3" [79] also lacks an explicit fitness function, instead using a social notion of fitness, implemented in a multi-agent system. In Kinetic Engine 3, individuals are selected for performance based on the appropriateness of the musical material they present in the given musical environment. If that material is deemed musically appropriate and useful to other agents, it survives, whereas if it is not, it fails to contribute to the musical development, effectively deeming it "unfit." This approach models human music improvisation, where material introduced by a given musician may fail to be developed during the course of the performance if it fails to capture the imaginations of the other musicians involved. Manaris and Roos present another interesting approach to the problem of assigning fitness [157]. In their system, a set of Artificial Art Critics (AACs) are used to model a social/critical notion of fitness. The AACs are implemented as neural networks that learn to evaluate a set of examples, based on a feature extraction process derived from power laws. In the course of their research the authors found that music estimated as "good" generally approximates a distribution close to that of so-called "pink," $1/f$ noise, when analyzed according to the metrics used. The AACs in their system are trained on a corpus of "socially sanctioned" music, downloaded as MIDI files from an online music archive, where quality is considered proportional to the frequency of downloads. Such an application of AACs as a fitness function for a genetic composition system provides an interesting and valid approach, even if the metric used for discriminating "good" and "bad" music is questionable.

From a composition theoretical perspective, perhaps the most compelling feature of EC is the emphasis it places on iteration as a fundamental creative strategy. The process of variation through reproduction and mutation can be considered analogous to the process of iterative refinement that characterizes human compositional practices [41]. Consequently, the incremental movement toward artistically satisfying solutions, through an accumulation of local, and often somewhat small-scale improvements, also provides a useful model of the relationship between musical materials generated *a priori*, and the continuous development of goals during compositional processes, as highlighted in Collins' theory (Section 2.3). This characteristic is somewhat unique to the EC approach, since the particular type of parallel and iterative search proposed has the potential to arrive at solutions immediately in some cases, while slowly refining solutions in others. This is particularly the case in

¹²The definition of the fitness function generally offers an opportunity for the designer/user to state explicit goals for the system's output. This is most clearly the case in interactive EC, where fitness is based entirely on the aesthetic evaluation of the user.

interactive EC, where well-formed solutions may not be apparent to the user until they are auditioned.

2.4.4 David Cope's "Music Recombinance"

David Cope's Experiments in Musical Intelligence (EMI) caused a sensation when it first came into prominence in the early 1990s, and has been viewed with a somewhat suspicious eye by the research community ever since [51]. The primary insight underlying Cope's success with EMI was his notion of "music recombance" (see Chapter 1), which asserts that new music is essentially a recombination and recontextualization of existing music, or more specifically, of the component structures of existing music. Early versions of the system [47] described the underlying formalism as an Augmented Transition Network (ATN) [246]. The ATN is an extension of Woods' Basic Transition Network (BTN), which can be seen as a graphical model of a formal grammar, with an added recursive function, allowing it to produce nested substructures within the larger generated sentence. The substructure produces a new graph, with its own root node, or "start point," and its own internal structure. During recursive generation, the BTN pushes the larger sentence structure onto a stack while the substructure (e.g., a prepositional phrase) is being generated. When the substructure is complete, the larger sentence structure is popped from the stack and generation continues. The ATN extends the BTN, by the addition of a test function and an action function. The test function determines whether a recursive construction may take place in the current context, and the action function provides a set of structure building rules if the recursive construction is permitted. Together, these functions allow the ATN to build complex sentences, potentially moving fragments of a sentence around, without sacrificing its deep structure (i.e., its fundamental semantic intent). According to Woods, a significant problem of context-free grammars that the ATN solves is the ability to generate structures in which sequential dependencies between non-adjacent parts of the sentence are retained.

Cope began his "experiments" with ATNs by writing simple programs to create haiku, following an intuition that producing reasonably intelligible and well-formed haiku was a task in some ways comparable to music composition. Subsequently, the principles of the ATN were applied to his notion of "non-linear composition" [47], in which quasi-linguistic function labels were assigned to phrases in a music database, which the ATN then used to build "parse trees" of new, syntactically and semantically valid works. For a given non-linear

composition process, a “source phrase” was provided by the user, and the ATN would proceed to expand this source phrase into a larger musical structure. In later versions of the system, the ATN approach was replaced with a more general pattern matching and search algorithm, based on the notion of music recombination. The exact process by which EMI composes new works appears to vary considerably, based on the formal tendencies of the style to be modelled, and the specific contents of the work being composed. However, the process could generally be described as a mapping of high-level statistical regularities from source works in the target style (the style to be modelled) onto the new work being composed. There are a number of attributes of the target style that EMI attempts to emulate:

1. Phrase length
2. Position of cadences (and distance between cadences)
3. Position and application of *signatures*, *earmarks*, and/or *unifications*
4. SPEAC structure

The first two attributes are self-explanatory; a style imitation B should generally match the phrase lengths and cadence positions of a target style A. But “signatures,” “earmarks,” “unifications,” and “SPEAC structure” need to be defined. Cope defines signatures as follows:

Signatures are contiguous note patterns which recur in two or more works of a single composer and therefore indicate aspects of that composer’s musical style. Signatures are typically two to five beats (four to ten melodic notes) in length, and usually consist of a composite of melody, harmony, and rhythm. [51]

In EMI, signatures are identified through pattern matching and, contrary to the general process of fragmentation that characterizes recombination, signatures are stored and applied intact. Of course, this is consistent with their definition, which specifies characteristic relationships between pitches and rhythms in a given harmonic setting, so that breaking them up would likely destroy the function of the signature. Earmarks are patterns which tend to precede, or “signal,” significant musical changes [51]. They generally appear near cadences, and at the end of sections, thus serving to “foreshadow important structural

events” [51]. Unifications represent the general strategy through which a musical pattern is applied in order to give continuity to a work.

While signatures contribute to the recognition of the style of the composer [...] unifications have greater local importance and relate to harmonic, thematic, and rhythmic elements only in a single work. [51]

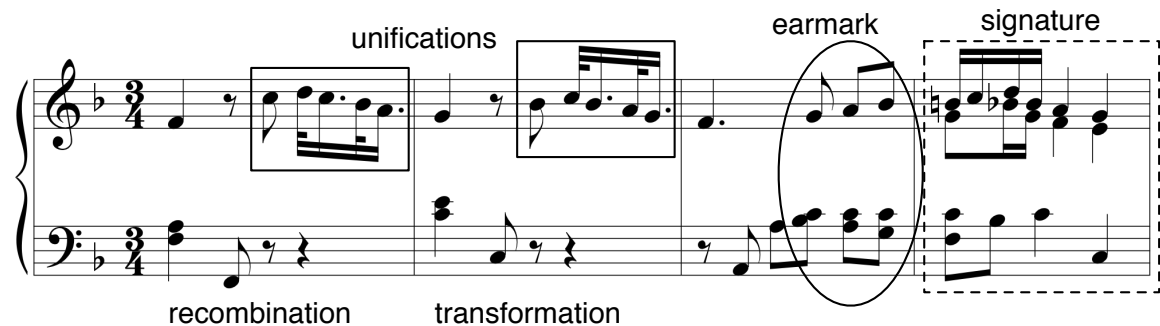


Figure 2.7: Output from EMI demonstrating various techniques, including “unifications,” “earmarks,” and “signatures” (reprinted from Cope [51]).

Unifications help provide continuity, through the local repetition or imitation of simple musical patterns; i.e., they serve to “unify” the local context. Figure 2.7 gives an example of how EMI exploits unifications, earmarks, and signatures in a compositional context.

Before discussing SPEAC analysis, we should introduce one of the primary objects in EMI’s music database: groupings. Groupings are vertical “slices” of music, containing the active notes across all voices of the musical texture at a given time, or over a given temporal window [51]. Groupings can range in length from a single onset to several beats in duration. In later versions of EMI, grouping length could be adjusted dynamically, as a result of the analysis process. Groupings represent the most basic units for recombination; simply put, an EMI composition can be viewed as a series of groupings from different source works, strung together in a novel order. When groupings are extracted from a source work, they are added to the database with references to the grouping which immediately follows them in the original setting. They are given generic names in the database, so that all groupings with similar attributes—for example, the pitch label {C-E-G}—will be stored at a common location. Part of the process of recombination involves chaining together lexically related groupings in a novel way. Of course, groupings are not always recombined exactly as they appear in the database. Transformational recombination [51] involves the alteration of a

grouping's attributes in order to provide a better "fit" in the novel context, or to allow for a greater variety of choices when the database provides too few options. Also, recombination does not always involve complete groupings, but may involve the selection of several melodic fragments from different groupings, which may require transpositional transformations in order to conform to the new harmonic setting. The details of such operations are beyond the scope of this introduction.

Cope's SPEAC system is a method of formal analysis, grounded in the idea that musical structure is communicated through changes in "tension" [51]. SPEAC is the technique that most closely connects later versions of EMI with the original ATN function implemented in the earliest versions. Using SPEAC analysis, a musical surface can be labelled according to the general formal functions of its constituent events (or groupings). The SPEAC labels—"Statement, Preparation, Extension, Antecedent, Consequent"—indicate the formal functions of the stored groupings. In SPEAC analysis, groupings are assigned SPEAC labels according to an analyzed degree of "tension." The attributes used for measuring tension are not prescribed, and could be any analysis properties associated with changes in musical tension: harmonic dissonance, "approach" tension (or voice-leading tension) [52], beat position, metrical position, and so on. All groupings in a given source work are assigned tension scores, after which they are sorted into five ranges, corresponding to the five SPEAC functions. SPEAC labels are assigned as follows [52]:

1. C → lowest tension groupings
2. A → highest tension groupings
3. P → groupings with tension closest to A
4. S → groupings with tension closest to C
5. E → remaining groupings

The main function of SPEAC analysis is to provide a strategy for resolving multiple, potentially conflicting choices among database items during recombination. For example, since many database items may exist with the identifying label {C-E-G} (i.e., a major triad), each with a unique contextual meaning in its original setting, SPEAC can help decide which {C-E-G} grouping might be most appropriate in the current context. Thus, if the context is a V chord in C Major, with an "Antecedent" SPEAC function, then a "Consequent" {C-E-G} triad might be most appropriate, whereas if the context is a tonic chord in G Major,

a “Statement” {C-E-G} triad may be better. Both are C Major triads, but in each case the structural implications, and subsequent database associations, will be quite different. Cope also regularly applied rewriting rules to SPEAC structures, treating the system as a context-free grammar in which deep SPEAC structures could be expressed through various alterations of surface structures. He never prescribed a set of specific rewriting rules, but rather gave the general guideline that SPEAC functions at higher levels should be expanded with lower-level sequences of functions that approximated their overall tension. An example set of rewriting rules is given in Table 2.4:

S	→	(P S)	(A C)	(S E)
P	→	(P E)	(E E)	(A C)
E	→	(S E)	(E E)	
A	→	(A E)	(P A)	(P A E) (S E A)
C	→	(C E)	(P C)	

Table 2.4: A set of rewriting rules for Cope’s SPEAC system (reproduced from Cope [52]).

Unlike the earliest versions of EMI, later versions composed in a somewhat more linear manner. The process began by selecting an introductory grouping from the database. Since the ending of this grouping would reference one or more other entities in the database as continuations, these continuations could be tested for compositional appropriateness. The choice of continuations was made with recourse to pattern matching procedures, SPEAC functions, target style phrase length, cadence positions, and so on, giving rise to an extremely complex decision process. In fact, Cope’s own analysis of a “Mozart” sonata composed by EMI [51] reveals the difficulty of easily quantifying EMI’s compositional process. Reading through this description, it is clear that Cope could only possess a general understanding of how EMI made the choices that resulted in the final composition, leading to an analysis reminiscent of conventional music theoretical analyses of the works of the Western canon. This is not an ideal state of affairs, and has likely led to serious misgivings on the part of the larger research community regarding the scientific validity of Cope’s work. Perhaps it was this general rejection by the research community that led to Cope’s highly controversial decision in 2003 to destroy the meticulously compiled music databases from which the thousands of EMI compositions were created [52]. Although the set of programs that make up EMI remain intact, Cope openly admits that the creation of databases for EMI compositions is a complex and highly selective process, without which

successful recombinant composition would be impossible. For this reason, it is a great misfortune that we can no longer refer to the databases, along with the software and musical output, without which we will likely never gain a more objective understanding of how EMI, with Cope as its database curator, achieved the remarkable success that it did. It is clear that the system is not embodied by the software alone, but rather exists between the software and the database(s) from which it draws its recombinant source material. Had these databases remained intact, it is at least conceivable that further research might have revealed the connection between EMI, its databases, the original source works, and the generated output, and perhaps even developed a way to reverse engineer the database creation process. As it stands, the details of EMI's success will likely remain a mystery.

From a composition theoretical perspective, Cope's work is of great interest, for the simple reason that it is a direct implementation of a composer's compositional instincts. Perhaps more than any other generative approach, Cope's notion of music recombination realizes the broad definition of composing as "collecting, connecting, placing, constructing, combining, and putting together several parts, through agreement or settlement, into a whole" (Section 2). Though it is clear that Cope's approach with EMI was rooted in linguistic processes, with the ATN serving a central role, knowledge of the ATN alone provides only limited insight into EMI's success. Although there is much discussion of statistical analysis [49], it is clear that conventional probabilistic Markov processes are also not central to EMI's design. Rather, statistical modelling is used primarily during evaluative procedures, in which the statistical regularities of productions (pitch occurrences, repetitions, phrase lengths, etc.) are compared to those of example works in the target style. In a similar manner, concrete predictions are not central to EMI's design, which is rather focused on modelling general patterns of expectation, as represented by the syntactic functions of the SPEAC identifiers. As a further contrast to contemporaneous systems, and again reflecting Cope's focus on modelling human compositional strategies, EMI's compositional process is highly iterative. Though not directly modelled on such systems, EMI's design is analogous to cognitive process theories of literary composition, utilizing a modular organization of several sub-programs, each handling a different aspect of the compositional process; analysis, pattern-matching, recombination, transformation, and so on [52]. EMI's compositional process proceeds via a complex negotiation of different (sub)programs, each of which fulfills a specific role in a generally non-linear series of compositional subprocesses. However, Cope's modelling of these processes does not follow the same organizational structure seen in models like that of Flower and Hayes. This is likely due to the fact that

EMI does not directly model human perception or cognition, but rather takes a music theoretical approach (i.e., to analysis, feature extraction, and so on). In this sense, it is very much an “expert system,” drawing on Cope’s compositional expertise, and his domain-specific knowledge of music theory, both of which are explicitly and implicitly coded into the system. Finally, although general hints and suggestions regarding database construction have been provided in Cope’s books, and several of his music analysis programs have been discussed in relation to database construction, the complete process has not been explicitly formalized or implemented in an algorithmic system.

Inspired by Cope’s Approach: The MusicDB

Cope’s approach to music recombination provided the basis for our “MusicDB” system, for MaxMSP [159]. The MusicDB is designed to facilitate recombinant composition, providing a method for analyzing MIDI files, and navigating the resulting data structure via a simple but powerful query syntax. The system employs Cope’s notion of Groupings and uses SPEAC analysis when analyzing MIDI files, building a hierarchical representation from one or more source works. Unlike Cope’s databases, which organize materials into “lexicons” based on the contents of Groupings, the MusicDB’s data structure is centred on SPEAC analysis, so that all of its musical content is accessed first and foremost by SPEAC function. Queries can search the entire SPEAC hierarchy, or can be isolated to specific branches of the hierarchy, localized by the results of previous searches. The analysis process breaks the input music into five primary descriptors: Events, Chords, Groups, Voice-Segments, and Phrases. Events are individual MIDI notes and Chords are collections of simultaneous Events. Groups are analogous to Cope’s “Groupings,” and Voice-Segments are short musical passages which may contain Events and/or Chords. Phrases are composite structures containing one or more Voice-Segments, the sequence of Groups that intersect those Voice-Segments, and all contained Events and Chords. The boundaries delineating different Voice-Segments and Phrases are determined through SPEAC analysis.

All descriptors are implemented as software objects with data fields defined for specific analytic qualities, and object referencing is used to define the relationships between the objects. When queries are made, the search is performed over all objects in the scope of the given search and data can be accessed either from the returned object, or from any object directly referenced by the returned object. Using this approach, it is possible to have different search and return types, so that one may use a series of Chords to search for a melodic interval sequence, or a rhythm to search for a series of Chords. However, as is

clear from the above discussion of Cope's work, the proliferation of data made available by such recombinant techniques can often lead to a similar proliferation of problems requiring solutions. Such is the case with the MusicDB where, in many cases, queries may lead to a wealth of information, with little guiding structure to aide in the recombination process. Extensive transformations are often required to adapt the extracted material to the novel compositional context, and this process must be carried out either by a separate compositional system, or by the user of the MusicDB. This is by no means a simple process, and promoted our continued search for a more integrated, holistic approach to music recombance.

2.5 If Composers are Creative, What Does That Mean?

Having discussed the models of compositional thought implied by various generative approaches, we will now look at a range of interpretations of what it means to *create*; that is, at what might be considered the ultimate goal of a generative system. Theories of creativity range from the philosophical and metaphorical to the formalized and quantifiable, but few, if any, cover the field in its entirety. In their overview of the subject, Kozbelt et al. [122] present a cross-section of prominent theories, framing their discussion according to a rule-of-thumb classification referred to as "the six P's of creativity." Briefly, this model proposes six general aspects of creativity that a given theory might address: *process*, *product*, *person*, *place*, *persuasion*, and *potential*. *Process* theories addresses the mental mechanisms and thought processes of individuals engaged in creative acts, while *product* theories address the products of creativity themselves, such as art works, inventions, publications, and so on. *Person* theories are concerned with understanding the personality traits of creative individuals, while *place* seeks to explain the role of geographical, cultural, and social context on creativity. Theories addressing the notion of *persuasion* bring to light the fact that creative acts and/or products are not always recognized as such by the larger public. Rather, it is often the case that influential individuals must persuade the public as to the value of a particular creative act or idea, in order for its creativity to be broadly accepted. Finally, theories of *potential* focus on types of creativity that might not be recognized as such, but that are distinctly creative, and might one day be recognized by the larger public. For the current discussion, aspects of *person*, *place*, and *persuasion*, while clearly of interest to understanding the phenomenon of creativity, are not of particular interest. Rather,

our focus will be on creative processes and products, and where appropriate, the potential for creativity that exploration in the area of process affords.

While any theory of creativity must obviously address the great creative thinkers of history—the Einsteins, Darwins, Mozarts, da Vincis, Shakespeares, and so on—it must at the same time account for creativity on a more modest scale. For example, the ability to express an idea in language is a capacity that virtually all human beings share. But it is also a distinctly creative act. We do not express ourselves by simply reciting valid sentences from memory, but rather combine words in novel, yet functional ways to communicate specific ideas in specific situations. Similarly, many individuals engage in creative acts like painting or writing, and often with a high degree of competence. But in the vast majority of cases, the results of these efforts cannot be classified on a par with the creative works of the masters. Aesthetic quality aside, however, it is clear that creative thinking must be involved. These more localized, everyday creative acts suggest that creativity should perhaps be understood in terms of its magnitude. To this end, many theorists have adopted a classification scheme proposed by Csikszentmihalyi [55], which accommodates the possibilities for “smaller c,” or “Larger C” creativity. Larger C creativity applies to the great achievements of humanity, like the general theory of relativity or the Mona Lisa, while smaller c creativity applies to the many broader, more commonplace acts, which would nevertheless be impossible without creative thinking.

2.5.1 Boden’s Creative Magnitudes and Types

Margaret Boden made an important contribution to the examination of creative magnitude by proposing the categories of “P-creativity” and “H-creativity” [24]. P-creativity, or psychological creativity, refers to ideas that are novel to the individual who created them. P-creative ideas need not alter the course of civilization; as far as the individual knows, they are original, and thus could not have been produced without some form of creativity. H-creativity, or historical creativity, on the other hand, refers to ideas that have been recognized as novel by humanity as a whole (or at least on a large scale), and have thus altered the course of history in some way. It should be noted that P-creativity necessarily subsumes H-creativity, since an H-creative idea must, at the very least, be novel to its originator. For this reason, Boden suggests that it is more important to understand P-creativity, even though the results of H-creativity may be more intriguing or appealing.

Extending these basic magnitude categories, Boden also proposes three categories of type: combinational creativity (C-creativity), exploratory creativity (E-creativity), and transformational creativity (T-creativity) [24]. Combinational creativity involves the generation of novel ideas by the association of disparate, or only tangentially related ideas. Thus, C-creativity proceeds fundamentally by analogy; the two (or more) ideas being brought together share some similarity of form, conceptual structure, or semantic content, and the novel idea reveals this similarity. The creative insight involved in C-creativity necessarily relies upon vast stores of knowledge, like that possessed by adult human beings. Significantly, this knowledge enables not only the generation of C-creative ideas, but also the appreciation of those ideas, since it is the comprehension of the underlying similarities that supports the meaning (and relevance) of the novel idea. As an example, Boden provides a short stanza from Shakespeare, which draws a series of analogies between sleep, mending a torn sleeve, a soothing bath, a balming ointment, and the second course of a meal, all in four lines of poetry. The feat of imagination and creativity on Shakespeare's part is clear, but what is perhaps not so clear is that any reader who comprehends and appreciates the stanza is likewise relying on C-creativity to do so¹³. Because of its reliance on knowledge, Boden suggests that C-creativity is perhaps the easiest for human beings to achieve, and conversely the most difficult for computer systems, which generally have limited, and overly specified knowledge structures.

E-creativity involves the exploration of a conceptual space for novel possibilities. An important aspect of such conceptual spaces is that they predate the individual creators that explore them; that is, they are cultural, social, and collectively defined spaces of ideas and possibilities. Examples of conceptual spaces include styles in the arts, theories in chemistry or biology, fashions or trends in couture, and so on. Such conceptual spaces are developed through the collective work of populations, and reflect the assumptions, values, and preferences of the culture (or, in some cases, another culture from which they are borrowed). Exploration of such spaces is thus always constrained by the parameters of the conceptual space itself, so that radical alterations or deviations are not permitted. More specifically, it is not necessarily a question of them being permitted, but simply that they will be rejected by the culture as a whole for failing to accurately represent the values of the group—which explains why exceptionally creative artists like Vincent Van Gogh, for example, may fail to achieve any degree of acceptance in their lifetimes.

¹³It is worth noting that Boden generally makes room for the creativity of the observer, showing that a degree of creativity is often required to appreciate other creative ideas. This is an idea developed by Peterson in the context of music, who looks at the creative role of the listener [188], as discussed in Section 2 (p. 9).

Even within a given conceptual space, however, there is often considerable room for exploratory creativity. For example, the sheer size of the conceptual space represented by a poetic style, like the “limerick,” makes it essentially impossible to explore the space of possibilities in its entirety. Thus, any individual who comes up with a novel possibility within the conceptual space—i.e., a novel limerick—is exhibiting E-creativity. In many cases, new E-creative ideas can lead the way for new ways of thinking within the conceptual space, sometimes opening up large and unsuspected areas for future exploration. Boden uses the metaphor of following a map on a country drive. The map represents the conceptual space which, while not a true representation of the open and unstructured geography of the terrain, nevertheless represents the structurally defined space of navigable roads and passages. One can depart from the freeway, and travel along backroads in an exploratory manner, without having any specific knowledge of where they will turn up. But it is always a certainty that the roads will lead somewhere, and that this “somewhere” can ultimately be located on the map. By this metaphor, we understand that it is the direction, path, and order of destinations that can vary—and on a map of sufficient size and complexity, the number of possible variations is staggering—but one must always follow a road or path, so that movement is always constrained.

Breaking the limitations of a predetermined conceptual space falls to the realm of transformational creativity. T-creativity takes some particular rule, formalism, or idea from the exploratory space and literally changes it, so as to enable the pursuit of some new possibility or space of possibilities. T-creativity is like following your road map to a known location, spotting some far off vista, and leaving the road to drive there directly. The promise of adventure is great, but so is the potential for failure. Thus, Van Gogh’s break with the conceptual space of naturalistic painting proved devastating to himself financially, but ultimately invaluable to the future development of the art form. An important point about T-creativity is that it literally enables one to think in ways that were not possible in the conceptual space prior to the transformation. Perhaps more significantly, because it opens up new avenues for thought, T-creativity also frequently leads to rapid developments in the field (generally new E-creative ideas) arising from the sudden expansion of the conceptual space. Perhaps not surprisingly, T-creativity is exceptionally difficult for humans to do. This is not always because there is anything particularly incomprehensible in the result. On the contrary, it is often the case that the T-creative idea reveals a possibility that seems beguilingly simple or transparent upon reflection; like the answer to a riddle. But however simple the alteration may be, the conceptual space has changed, and would not have changed without the

advent of the T-creative idea. As an example, the musical spaces opened up by Reich's minimalism seem self-evident now, in spite of the fact that they were anything but obvious at the time Reich introduced them to the musical public. This is not surprising, since the conceptual space Reich inherited from the modernist avant-garde was so constrained, distorted, and over-determined as to make his approach to form seem almost nonsensical at first. And yet, once the conceptual space opened up by minimalism began to be explored by the musical community, it flourished to become one of the most influential compositional approaches of the twentieth century.

2.5.2 Schmidhuber's Theory of "Compressor Improvement"

Jürgen Schmidhuber presents a compelling formal model of creativity based on information compression and intrinsic reward for compressor improvement [208]. The model includes a compressor/predictor representing an adaptive world model, a learning algorithm that attempts to improve the world model, a system of intrinsic rewards for improving compressor performance, and a reinforcement learner that tries to select actions that will optimize future reward. The underlying idea is that, because human beings live in an information-heavy world, learning to detect order in that world is implicitly rewarding; crudely stated, we enjoy learning to do things, and our brains enjoy learning to simplify information we encounter. This process of simplifying representations can be described as a form of compression. As Schmidhuber explains:

Newton's law of gravity can be formulated as a short piece of code which allows for substantially compressing many observation sequences involving falling apples and other objects. [209]

This notion of compression is a formal concept, and is framed specifically in terms of computational models, but the fundamental point is that the concept of gravity (whether defined formally or not) allows us to accurately predict the immediate future of objects dropped, thrown, knocked off ledges, and so on. Beyond the specific case of gravity, there's also a certain utility in learning to predict environmental events; one can save oneself a great deal of confusion, wasted effort, or even serious physical harm (and death), by learning to better predict events in the world. Thus the intrinsic motivation of learning to understand the world can also be seen as a strategy for improving external reward, by extending the life of the organism.

Beauty in Schmidhuber's model is proportional to the compressibility of the information. For example, faces that are highly symmetrical, maintain fairly normal proportions, and are more similar to very familiar faces tend to provide the standard for beauty. In a similar manner to Schmidhuber's example of the theory of gravity, these are faces that compress many observations of other faces into "a short piece of code," representing an optimal face. However, Schmidhuber is quick to point out that this does not mean that such faces are interesting. Interestingness, unlike beauty, is proportional to the ability of the compressor to further compress the information; that is, it is linked to the intrinsic reward received for compressor improvement. If a face is too conventionally beautiful, we lose interest quickly. Thus, faces that are highly symmetrical, have fewer unique features, and are familiar can only be interesting to the degree that all of these attributes are moderated by some kind of uniqueness; that is, by the presence of information that cannot immediately be represented in an optimally compressed form. It is the desire for comprehension—the striving for compressor improvement, which Schmidhuber refers to as the "learning curve"—that is intrinsically rewarding, and therefore maintains our interest. Considered in the context of music the model suggests that music with enough familiarity and regularity to be considered beautiful, but enough novelty to challenge the compressor, will be most rewarding for the listener. Thus, music that is overly familiar, or overly simplistic, given the listener's knowledge and experience, will soon become boring, just as music that is overly complex or irregular, with no perceivable pattern, will also become boring. As a final note, it is worth mentioning that Schmidhuber's model, being focused more on a process (i.e., "compressor improvement") than a product, can also directly account for the creativity inherent in perception, as discussed in Peterson's examination of creativity in music listening [188].

2.5.3 Graeme Ritchie: Quantifying Computational Creativity

Graeme Ritchie's work on creativity theory emphasizes the evaluation of creativity, with a particular focus on assessing the potential for creativity in computer systems [197, 198]. Similar to Schmidhuber, Ritchie gives a relatively simple, quantifiable definition of creativity, founded on the estimation of three properties: *novelty*, *quality*, and *typicality*. In Ritchie's definition, novelty is the extent of a produced item's dissimilarity to a given class of items, while quality is the extent to which a produced item can be considered an example of that class [198]. Typicality is slightly different, in that it takes into account the "artifact class" of the item. Generally speaking, in human evaluations of creativity, it is assumed that the item being evaluated is a member of a known class of artifacts; e.g., songs, stories,

poems, pictures, and so on. Typicality, then, concerns the extent to which a produced item is an example of such a known class. Ritchie acknowledges, but does not directly address, the rather important issue of distance/similarity metrics, which remain an open question in music research. However, the formalism presented by Ritchie does not concern the quantification of such comparisons, since his stated assumption is that evaluations of creativity are generally made by human subjects. This is not to say that machine evaluation is impossible, but simply that the mode of evaluation is not important, only the estimation of novelty, quality, and typicality. Given a set of judgements (human or machine) based on these attributes, Ritchie's formalism will provide an overall estimate of creativity.

Unlike Schmidhuber's approach, which is not specifically concerned with categories (as in Boden, for example), Ritchie's approach is fundamentally grounded in set comparisons. He provides a comprehensive list of "criteria" calculations, which give $[0, 1]$ evaluations of the estimated creativity of a given set of productions R , based on comparison to the set of all members of the artifact class B , and a set of "inspiring" artifacts I , which represents the implicit or explicit assumptions of the program designer regarding the formal properties of the artifact class (which have potentially guided the development of the system). I can represent a set of specific sample works, as in corpus-based machine-learning systems, or the set of rewrite rules defined by a formal grammar. Interestingly, I can also be empty, as in mathematical models, or other "sonification" approaches, which make no assumptions about musical structure. Ritchie makes the important observation that a system capable of replicating an item from I —for example, literally quoting from the corpus—though not particularly creative, may nevertheless be useful, as its implementation may reveal important information about the formal structure of I , or the creative process that gave rise to the class. It is also worth noting that Ritchie's formulation of I , as a sort of "meta-class" of creative items, can also be said to represent the composition theoretical assumptions of the system's design. For example, a system based on simple Markov chains only assumes that a given class of compositions can be represented by a probability distribution, and that generation is a predictive process, as was suggested in Chapter 1. If such a system uses a 1st-order Markov model induced from a given corpus (i.e., rather than being coded by hand, as in Xenakis' early work with Markov models), then, as was shown in the simple example of the Mozart melody, the probability of reproducing an item from I is vanishingly small. Such an incapacity to represent I could thus indicate a weak theory of composition.

Creativity and Style Imitation

As a sort of epilogue to this discussion of creativity, the question is often asked why Cope, or any other researcher studying computational creativity in music, would focus on style imitation; i.e., in teaching a computer to compose “in the style of” Bach, Mozart, Chopin, and so on. The most straightforward answer is that style replication provides a solid ground for assessing the “quality” side of the novelty/quality/typicality formula. If a system can convincingly compose in the style of a given composer—often times a composer already noted for H-creative musical contributions—without merely replicating existing works, then that system must at least be capable of some form of P-creativity. Further, in consideration of the many applications of generative systems in the field of CAC, it is reasonable to ask that a compositional “collaborator” have enough skill in the field to produce convincing results, that are at least recognizable as such by the user of the system; i.e., that are successful in terms of typicality. Often, researchers in the development of creative systems focus too quickly on H-creativity, ignoring the fact that it is often difficult, if not impossible, for individuals to assess the H-creativity of a given artifact, the necessary information simply not being available to the individual assessor.

Chapter 3

Music Perception and Cognition

In his book *Music and Memory* [216], Bob Snyder provides an overview of research in music perception and cognition, introducing central theories about the memory structures and cognitive processes involved in understanding music. It is important to acknowledge that much of Snyder's discussion is "transcription-based" or "note-based" [205], revealing a clear intention to connect the perceptual and cognitive functions of audition to the higher-level capacity for musical and compositional understanding. This approach, while potentially biased from the purely psychological point of view, is well-suited to our work, which also focuses on relating concepts from the music perception and cognition literature to the highly specialized act of music composition. A comprehensive discussion of the literature is beyond the scope of this dissertation, but the following chapter will provide a brief overview of those concepts that are directly relevant to our work.

3.1 Echoic Memory

Snyder's discussion begins by addressing the earliest stages of auditory processing, in a short-term memory function referred to as "echoic memory." The primary role of echoic memory is to allow an acoustic stimulus to persist in memory long enough that it may be subject to processes of "perceptual binding" and "feature extraction" [216]. Through these processes, vast quantities of acoustic information—impulses emitted by thousands of nerve cells in the ear—can be categorized and "grouped" into basic percepts, allowing the brain, for example, to "bind" a set of simultaneously sounding frequencies into a singular perception of "pitch." Extensive theoretical and experimental work has been done on the subject of echoic memory, and computational approaches have been proposed

[56, 106, 117, 145, 182]. However, because the generative music systems we are discussing here deal with symbolic music representations, low-level functions like pitch perception and onset detection are treated as *a priori* requirements, and will not be discussed in detail.

There are, however, a few key aspects of echoic memory processing that are directly relevant to our work on MusiCog; in particular, the notion of “primitive grouping.” Perceptual binding and feature extraction transform temporal processes into discrete percepts, allowing us to identify the various actors and forces in our immediate auditory environment—a speaking voice, music, traffic in the street outside, water pouring into a glass. However, in order to derive meaning from such auditory streams, we must first become sensitive to qualitative changes in the streams over time.

When some aspect of the acoustic environment changes sufficiently, a boundary is created. This boundary defines where a grouping begins or ends and is the most basic kind of feature that is detected in the earliest stages of perception. [216]

Such boundaries are perceived via a phenomenon known as “closure,” which gives groupings a sense of separation and identity, transforming the continuous flow of acoustic information received by the ear into a sequence of meaningful auditory events. Primitive grouping thus provides the foundation for the higher-level cognitive processes that allow phonemes to be combined into words and words into sentences, or individual pitches combined into motives, phrases, and melodies in music. A sudden change of pitch, or the separation of events by a period of silence, both provide strong sensations of closure, and are thus frequently cited as mechanisms for primitive grouping [30, 185, 216]. Significantly, primitive grouping also appears to be an innate capacity of the auditory system, functioning independently of acquired musical knowledge or high-level cognitive processing.

Closure need not be absolute, grouping all features of an auditory stream equally. Rather, different features of the stream—pitch and rhythm, for example—may suggest different points of closure; a phenomenon referred to as “partial closure” [216]. In partial closure, a pitch boundary may be detected independently of a rhythmic boundary, or vice versa. Stronger sensations of closure, and thus more prominent boundaries, tend to be characterized by partial closures across multiple features; e.g., simultaneous pitch and rhythm closure. It is worth note, however, that rhythmic closures, particularly those involving periods of silence (i.e., “rests”), tend to dominate the overall perception of closure in

melodic contexts [184, 225], such that pitch transitions that would not otherwise be perceived as boundary forming may be perceptually separated by the introduction of rests or significant alterations of rhythm. Boundaries may also be suggested by changes involving other features like loudness (i.e., dynamic accents), pulse or “tactus”, and metre.

3.2 Auditory Stream Segregation

An important cognitive phenomenon arising from the low-level categorization of acoustic information in echoic memory is the formation of auditory *streams*. A detailed discussion of auditory stream segregation is beyond the scope of this dissertation. However, some understanding of this fundamental idea is vital to any account of the cognitive roots of music composition, since the conceptual foundations of many compositional practices arise as a direct result of the phenomenon of streaming. As discussed in Section 3.1, the formation of individual auditory percepts in echoic memory supports the identification of forces and actors in our surrounding environment. In general, these forces and actors, as physical entities in the real world, maintain a degree of continuity through time. For example, real-world things do not move instantaneously from one spatial location to another; rather, they transition continuously between locations, in a generally predictable manner. Thus, when two successive sounds do come from completely distinct directions, it is likely that they have two separate causes. Over the course of evolution, the auditory system has become attuned to the fact that sounds maintain certain statistical regularities through time, and that these regularities can be used to identify the sources of the sounds themselves. Sources of harmonic, or pitched sounds, for example, generally have physical properties that constrain the specific mixture of frequencies—or *spectra*—they produce, so that sounds with dramatically different spectra are also likely to come from different sources. It is a similar case with the production of sequences of pitched sounds. Voices, for example, do not transition instantaneously from one pitch to another, but rather transition in a continuous (if rapid) manner. Over millennia, the auditory system has become sensitive to the fact that pitch sequences with a consistent register (i.e., with relatively small differences in fundamental frequency) are likely to come from the same source. Over time, continuity of spatial location (“localization”), spectral structure (“timbre”), and pitch have come to be used as auditory cues for building a predictable mental model of the forces and actors in the physical environment.

In music, auditory stream segregation has played a central (if unconscious) role in the exploration of form and structure. The constraints imposed upon basic formal types like homophony and polyphony, as well as many techniques in the field of orchestration, can be understood as direct responses to the subjective experience of auditory streaming, and in some cases the desire to expand and challenge our experience of it. Huron has shown [105] that the common voice-leading rules outlined in contemporary harmonic theory texts—i.e., avoidance of voice crossings, avoidance of melodic leaps, avoidance of parallel motion, etc.—can be explained as the intuitive optimization of compositional practice in support of auditory stream segregation. Further work by Cambouropoulos [32] extended these findings to the field of computational voice-separation. Starting from an interrogation of how the term “voice” is commonly understood in music theory, Cambouropoulos noted that single monophonic lines, in many cases, would outnumber perceptual streams, due to processes of “onset synchrony” (notes occurring at the same time), “co-modulation of pitch” (in music theory, “parallel” or “similar motion”), and “vertical integration” (the perceptual fusion of harmonically related pitches—a phenomenon of *grouping*). Musical passages exhibiting all of these properties could lead to the illusory perception of a single musical line in cases where multiple monophonic voices were actually involved. The most obvious case is the orchestral practice of “doubling,” where a single line is played by two different instruments. When done expertly, doubling can produce a sound quite distinct from that of either instrument heard in isolation, giving a strong impression that a single—and potentially unfamiliar—instrument is playing. In other contexts, stream segregation can produce a perceived multiplication of voices, when only a single voice is present. This phenomenon, referred to as “virtual polyphony” [248], has been exploited extensively in music, often by introducing unexpected pitch leaps in the context of rapid, isochronous rhythmic values. An example of virtual polyphony is shown in the excerpt from Bach’s E Major Partita for violin solo, BWV 1006, in Figure 3.1. Here the close pitch proximity and repetition-based expectancy produced by the upper figures form one stream, while the slow descending passage (A4, G \sharp 4, F \sharp 4) invokes the perception of a separate stream below.

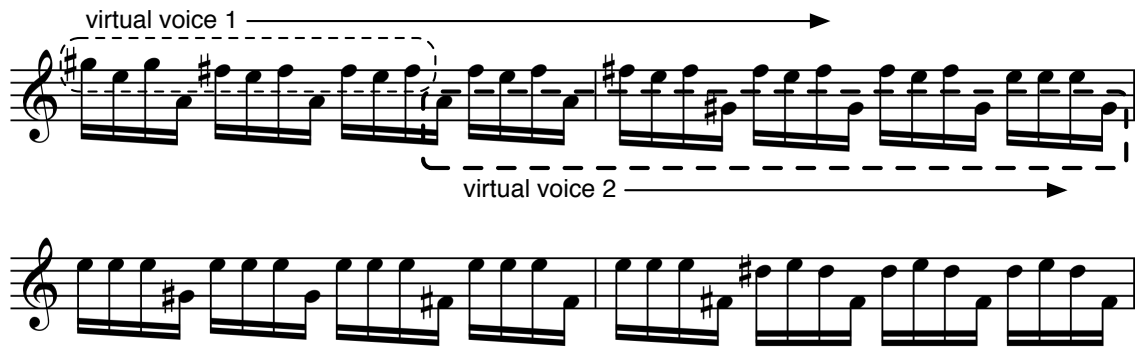


Figure 3.1: An example of “virtual polyphony” in the Preludio from Bach’s E Major Partita for violin, BWV 1006.

3.3 Short-Term Memory or “Working Memory”

Once echoic memory has categorized acoustic stimuli into basic auditory objects, an immediate interaction with higher-level cognitive processing takes place, through which long-term memories are said to become “activated” [216, 245]. Significantly, these interactions between echoic and long-term memory (LTM) appear to be pre-attentive, and have been shown to be influenced by expert knowledge, so that, for example, trained musicians are more discriminating of musical sounds in both attentive and non-attentive situations than non-musicians [118]. The long-term memories activated by echoic memory are not necessarily all brought into conscious awareness in a uniform manner. Rather, those memories that are more highly activated tend to be more easily retrieved, and are thus considered to be immediately accessible to conscious attention. This temporary memory store, which has been supported by associations in LTM and is thus immediately accessible to consciousness, is referred to as short-term memory (STM). Similarly to echoic memory, STM is quite limited in its storage capacity, but unlike echoic memory, which retains raw auditory percepts, STM deals primarily with categorized auditory objects. STM is generally considered to be a memory *process* (rather than a discrete memory *system*), which appears to operate over a number of different types of information, associated with different sensory modalities. An important property of STM is that it preserves the serial order of experiences in our immediate past, thus helping to build a relatively stable mental representation of the world (in spite of the fact that our conscious perceptual experience is actually quite fragmentary and discontinuous). As such, the ability to extract and follow the semantic

structure of event sequences, so essential to understanding musical discourse, is quite unimaginable without the functioning of STM¹.

Like echoic memory, there is a limit to the number of unique items STM can store at any given time. Whether this is truly a “hard-wired” capacity limit, a function of time or duration, or some side-effect of attention, interference (i.e., with other concurrent information), or other cognitive factors, is a topic of continuing debate in the research community. Nevertheless, experimental and theoretical work suggests that some sort of capacity limit does appear to exist, though it is more commonly associated with a limit on the number of “chunks” (see Section 3.4) that can be retained, rather than the number of individual items [53]. Where STM duration is concerned, the experimental values are generally similar to the limits of echoic memory, showing decay around 2-5 seconds, but extending to as long as 10-12 seconds under certain conditions [216]. In order for short-term information to be retained for longer periods, some form of “rehearsal” is required to keep memories activated. This rehearsal may involve the conscious, deliberate repetition of items, or it can be more subtly enacted. For example, the common rhetorical device of emphasizing a point by repeating its central idea or concept, often through the use of synonymous verbal structures (i.e., the individual items are different, but the associated ideas are the same), is a literary/oratorical form of passive rehearsal. In the case of music, this is often achieved through repetition and imitation, by which composers are able to support a musical idea in the listener’s memory without the listener necessarily becoming consciously aware of the rehearsal process.

A more general short-term model referred to as “working memory” (WM) is used to identify the broader collaboration of echoic memory, STM, and LTM, while adding a separate processing element required for carrying out cognitive tasks. Perhaps the most popular and influential model of WM has been proposed by Baddeley [12]. This model includes an executive controller and two slave systems: the “phonological loop” and the “visuospatial sketch pad.” The phonological loop itself consists of two parts: a temporary information store and an articulatory rehearsal process, similar to subvocal speech [13]. The temporary store is limited not by the number of unique items, but rather by the duration over which any single item can be stored, so that rehearsal is required to maintain items in the store. However, because rehearsal is proposed to take place in real-time, items will often be dropped from memory before the rehearsal is completed. The visuospatial sketchpad was proposed as

¹In fact, short-term memory deficits have been shown to be a factor in *congenital amusia*, an impairment in the processing musical information [244].

a system for dealing with visual information in a way that made an important distinction between the appearance of objects and the spatial relationships between those objects. Logie [150] proposed an extension to Baddeley's original model, which incorporated a "visual cache" providing a similar function to the temporary store in the phonological loop; i.e., allowing visual information to be temporarily stored for rehearsal, manipulation in the visual imagination, and so on. Similarly, Berz [19] proposed an extension to Baddeley's model, which introduced a separate "music memory loop," potentially attached to the phonological loop, and allowing for the processing and storage of musical stimuli separate from verbal and visual information.

3.4 Memory Optimization and "Chunking"

Given the rather modest capacity of STM/WM it seems that some additional mechanism must support our ability to quickly store and recall relatively long streams of information. This mechanism, referred to as "chunking," is a form of compression that collects associated information into groupings that can be recalled as unified items or concepts. Snyder gives the example of trying to recall the numeric sequence: 1776148220011984. A naïve, "brute force" attempt will prove quite challenging, since the information is presented as a sequence of 16 seemingly random digits. However, once it is realized that the sequence is actually a series of dates, the information can be "chunked" into only four elements, greatly reducing the cognitive effort required for storage and recall. In part, this is due to the fact that dates have a familiar formal structure (i.e., a 4-digit value, generally less than or equal to the current year), but also because they are often included in still higher-level chunks, associating information about historical events, personal experiences, and so on. In the domain of music, a similar problem is posed by the task of recalling moderately long melodic passages. The opening theme from Mozart's 40th symphony, for example, comprises a 20-note sequence. Based on STM capacity alone, it should be quite challenging for a listener to maintain a mental representation of this structure during listening. However, when the phenomenon of chunking is combined with the previously discussed notion of grouping via closure, the cognitive demand of the task is greatly reduced. Taking grouping into account, this 20-note sequence can be reframed as a series of six melodic chunks: (E \flat , D, D), (E \flat , D, D), (E \flat , D, D, B \flat), (B \flat , A, G), (G, F, E \flat), (E \flat , D, C, C). It is worth note that this particular passage also exploits imitative compositional techniques, further aiding storage and cognition (i.e., via passive rehearsal), as discussed in Section 3.3.

Another important property of chunking is that it is recursive, so that repeated exposure to information will facilitate the construction of higher-level chunks, as sequences (or collections) of lower-level chunks. This process results in further memory optimization while also enabling the development of hierarchical knowledge structures. When learning and recalling longer musical sequences, chunks are connected together via “cueing”; a process in which a chunk’s ending becomes the “cue” for another chunk. It is worth noting, however, that although this process helps support storage and recall, the elements within the stored chunks become difficult to access individually, without reference to the original sequence. Thus elements at the chunk’s “boundary” tend to be easily recalled [25, 216], while other elements must be recalled by stepping through the original sequence, beginning at the boundary. For this reason, elements located in the middle of chunks tend to become difficult to recall without reference to the complete sequence. The structures defined by chunks, aside from aiding in the storage and recall of sequential information, also provide a degree of conceptual abstraction. Chunks can be regarded as mental concepts in their own right, with certain general properties and structural relationships to other chunks, and often to associated items of declarative knowledge like titles, composers, styles, historic periods, techniques, and so on.

Language in Music Psychology, Music Theory, and Composition: Clarifying Terms

As should be clear from the previous discussion, the cognitive process of chunking leads to the formation of musical *concepts*, which possess a degree of abstraction. In a certain sense (and to a certain degree), they can be contemplated independently of time, and treated as constitutive units in a kind of grammar. These units have been given names in the musical vernacular like “motive,” “klang,” “phrase,” and so on, and their structural and relational properties have tempted many theorists to draw analogies between music and natural language [18, 47, 183, 190]. This association has led to a fairly widespread borrowing of linguistic terms in the field of music psychology, without always taking care to clarify the musical sense of the meaning. The term “musical surface,” for example, appears frequently in discussions of the cognition of musical form, but the significance of the term is not always made clear. As it is generally applied in music, the term is synonymous with a related idea from linguistics (via Chomsky [?]), and refers simply to the musical sounds as they are experienced in time (or indeed, as they are performed by a musician, or read

from the printed page). Of course, this seems obvious, since sounds are clearly the tokens of musical communication. But in the context of Chomskian linguistics such surface events must be considered in relation to higher levels of so-called “deep” or “background” structure, as is the case in Schenkerian analysis or the structural formulations of Lerdahl and Jackendoff’s GTTM [146]. Thus, when a music psychology study highlights a listener’s ability to extract information from the “musical surface” alone, there is often a connotation that a significant influence from higher-level cognitive functions is also involved, enabling the inference of structural features with long-term temporal dependencies; a common example is the induction of tonal hierarchy. In this sense, the use of the term, considered for its special connection to linguistics, carries a meaning that might not be made clear through the use of more pedestrian language.

In a similar manner, the term “musical parallelism” is often used in place of the simpler, or at least more familiar term, “similarity.” Like the notion of the musical surface, the term parallelism also comes to music psychology discourse via linguistics; or more precisely from grammar, language, and rhetoric. In writing and speech, parallelisms are used to support grammatical or rhetorical structure, improving written (or verbal) communication and aiding comprehension [22] (i.e., through a form of passive rehearsal in WM, as outlined above). For example, the mixing of gerunds and infinitives is an error of parallelism that is considered poor writing style, and leads to confusing statements; e.g., “Like most composers, he enjoys drinking, carousing, and to sleep.” A correct *parallel* construction would use only gerunds and would read: “Like most composers, he enjoys drinking, carousing, and sleeping.” In rhetoric, on the other hand, parallelism refers to the use of structural and/or semantic repetitions in order to provide emphasis. Quoting Stravinsky: “Cacophony means *bad sound, contraband merchandise, uncoordinated music* that will not stand up under serious criticism” (parallelisms in italics). The important point here is that the parallelism, though grounded in a certain type of similarity, does not require *surface* similarity; i.e., the words do not *resemble* one another. Rather, the similarity is found at a higher, or more abstract, structural or semantic level. In music, parallelism is difficult to quantify, as it may be displayed through multiple modalities—e.g., pitch and/or rhythm—the interdependencies of which are not entirely understood. Pitch contour, due to its efficacy for short-term melodic identification [187], tends to be an effective source of parallelism, but it is by no means the only such source. It is worth note that Cope’s notion of “unifications” (Section 2.4.4, p. 42) is in many ways a description of musical parallelism.

3.5 Cognitive Modelling

Purwins et al. [193] provide an overview of existing approaches to computer modelling of music perception and low-level music cognition. Building on this foundation, they go on to present a number of systems [192] that can be used to explain higher-level music theoretic concepts like *rhythm*, *beat*, *musical voice*, *consonance*, and so on. One important outcome of their survey is its recognition of the need for an overarching theory of how the different aspects of music cognition—and by extension, the various systems that model these aspects—combine and interact during the cognitive processing of music.

A number of cognitive theories of musical form can be described as *generative*—i.e., they are both *descriptive* and *prescriptive* theories. Lerdahl and Jackendoff's GTTM [146], Narmour's Implication-Realization (I-R) model of melodic perception [175], and Desain's (De)Composable Theory of Rhythm [60], are a few examples. These theories are not systems used to produce music directly, but rather are generative in the Chomskian sense that they characterize the kind of knowledge an agent must possess in order to understand and express musical ideas. The formalization of such knowledge in a system can become extraordinarily complex—which may explain why there is, as yet, no direct implementation of the GTTM. Further, because such models are often grounded primarily in music theoretical knowledge derived from the analysis of existing musical works, they tend to be characterized by exceptions, preferences, and special case rules, added to maximize applicability and generality in the face of the vast array of subtly differing contexts encountered in actual music.

3.6 Cognitively-Grounded Music Generation Systems

Although the use of cognitive models for autonomous music generation is relatively rare—the majority of systems function as tests of specific computational formalisms—a number of interesting approaches have been explored. We will introduce three systems here.

3.6.1 The IDyOM Model

Wiggins et al. developed their IDyOM (Information Dynamics Of Music) model [242] with the specific intention of using an empirically validated cognitive model of listening [186] for the generation of chorale melodies, under the general assumption that “in order to compose music, one normally needs to learn about it by hearing it” [242]. IDyOM is a statistical

n -gram model, based on the multiple viewpoint approach of Conklin and Witten [45]. It operates on two separate time-scales, utilizing a short-term model and a long-term model. The short-term model builds an incremental statistical representation of event sequences, as they are encountered during prediction, whereas the long-term model is trained on a corpus of source works, and is intended to represent the statistical regularities demonstrated by an entire genre [186]. Both models work together in the process of forming predictions.

Features used for the unsupervised training of IDyOM were chosen based on their ability to maximize predictive success during listening. Composition in IDyOM involves the extraction of a pitch distribution from the state space of the trained model using stochastic sampling techniques. Beginning with a set of 7 corpus melodies (which had been excluded from the training set), 7 novel melodies were generated using 5000 iterations of Metropolis sampling; a process that stochastically replaces the sample melody events with events from the statistical model. Wiggins et al. are careful to point out that they do not suggest that such a stochastic sampling process represents human compositional thinking, but simply that it helps to produce an unbiased sampling from the statistical model². Evaluation of IDyOM's output, by a panel of 16 experts using Amabile's "Consensual Assessment Technique" [5], revealed that the model was incapable of consistently generating convincing chorale melodies [242]. Although there are many possible reasons for the model's limitations, a number of which are given by the authors (the Metropolis sampling approach, features selected for training, etc.) [242], the results do throw into question the general notion that listening alone provides sufficient knowledge for composition. Or, if we accept this notion, we must question whether a successful model of melodic prediction can genuinely be considered equivalent to a successful model of listening. By deduction, we might ask whether there is any reason to equate stochastic selection from a model of musical prediction with the kind of understanding required for composition.

In Figure 3.2 we provide a manual segment/phrase structure analysis of IDyOM's most successful chorale melody (as estimated by the listening model). The example reveals a general weakness of IDyOM's output in the areas of motivic exploitation and parallelism, and repetition structure; features that strongly characterize the original Bach chorale melody.

²In this sense IDyOM is not specifically a cognitive model of composition, but rather a cognitive model of listening. Compositionally, the model suggests that new music is fundamentally a stochastic sampling from a knowledge representation of previously heard music.

J.S. Bach *Jesu, meiner Seelen Wonne*: "Base melody" for IDyOM Generation

IDyOM System D Melody (base melody: *Jesu, meiner Seelen Wonne*)

Figure 3.2: Comparison of motive and phrase structure in the Bach's *Jesu, meiner Seelen Wonne*, and IDyOM's most successful generated melody using the Bach as a "base melody."

3.6.2 The Anticipatory Model of Cont et al.

The "Anticipatory Model of Musical Style Imitation" (AMMSI) of Cont et al. [46] is designed to model the role of expectation in music, looking specifically at the notion of anticipation. The authors define anticipation as "the mental realization of possible predicted actions and their effect on the perception of the world at an instant in time" [46]. Like IDyOM, the model uses a multiple-viewpoint representation, but rather than storing long-term knowledge in a statistical n -gram model, it is represented using a pattern-matching data structure and algorithm called the Factor Oracle (FO) [4]. Because a central focus of AMMSI is to model musical interaction in an improvisatory context, the FO is an ideal representation, as it builds its data structure incrementally, in real time. The FO can be traversed in various ways, by following "links" in the data structure, and can be rapidly searched for a context

matching that of an input musical sequence. The FO is the generative formalism used in the well-known “Omax” system for MaxMSP, by Assayag et al. [9].

AMMSI uses multiple, situated agents whose environment is represented either by live musical input, an incrementally presented score, or feedback from its own output. The system learns to adapt to its environment using a Reinforcement Learning (RL) system. RL is grounded in the notion of assigning reward for successful interactions between an agent and its environment, and the goal of the RL learner is to optimize its received reward over time. In AMMSI, reward is associated with the appropriate estimation of the agent’s musical context (i.e., the environment), and the generation of appropriate musical materials, given that context. Reinforcement signals in AMMSI are used to guide the system toward states in the memory model (the set of FOs) that are most productive for learning and generation. Cont et al. suggest that this represents a form of attention direction, focusing the agent’s processing on the most appropriate locations in memory. However, because the FO, as a generative model, is prone to falling into repetitive loops if a given memory location is directly supported by current input, AMMSI also employs a suppressive reward scheme when responding to feedback input. A particularly interesting aspect of AMMSI’s design is its use of a competitive and collaborative approach to model selection. Similar to Conklin and Witten’s earlier work with multiple-viewpoint context models, AMMSI uses predictive success to guide the selection of agents and FOs during learning, assigning the RL reward signal to the best performing agent.

Due in part to the generative power of the FO, which stores all musical information in its training corpus, in its original sequential order (and is thus arguably a more strictly *re-combinant* system than IDyOM), AMMSI is a fairly successful generative model. Analyzing the sample generation provided [46] we noted that the system demonstrates a degree of motivic exploitation, and also an ability to reuse phrase-level structures. Somewhat uncharacteristic syncopations do appear in the output, but are generally short-lived, returning to more characteristic rhythmic patterns with acceptable regularity. It is worth noting, however, that AMMSI does not specifically attempt to model melodic composition, but rather to model improvisation. This is by no means the same task, as melodic composition generally involves motivic development and the promotion of closure (see Section 3.1, p. 3.1), whereas improvisation is, broadly speaking, focused more on elaboration and exploration of a musical space.

3.6.3 A Deep Learning Approach

Another area of active investigation is the use of so-called “deep learning,” and “deep belief networks” (DBNs) for music classification, prediction, and generation (for an overview of deep learning, see [16]). Deep learning systems are connectionist models that attempt to construct multi-level, hierarchical representations of a given input space. The goal is to untangle the many complex, non-linear relationships between simple features of the sensory environment and the vast number of combinations of those features that constitute a world model. Deep learning proposes a method to discover, for example, the complex combinations of simple shapes that combine in cognition to form a face, or the relationships between individual sounds, notes, melodies, and so on that combine to form high-level abstractions like “song,” “symphony,” or “Mahler’s 5th.” Jeff Hawkins’ Hierarchical Temporal Memory model [98], which served as the basis for our earlier work on the Hierarchical Sequential Memory for Music [161], was essentially a deep learning model, though the implementation used a novel Bayesian approach. Although deep learning does not seek to model individual cognitive functions (i.e., from the psychology literature; perception, STM, LTM, declarative knowledge, and so on), dealing rather with relatively high-level modelling of neocortical functions, its direct connection to neuroscience makes it highly relevant to the current discussion.

As yet, there are only a few examples of the application of deep learning to music generation. One example is the generative model of Bickerman et al. [20], which uses a DBN for learning and generation of Jazz melodies. It should be noted, however, that this model is designed to follow a given harmonic outline, and is thus perhaps more closely related to Allan’s work with HMMs [3] for chorale melody harmonization (see Section 2.4.1). Obviously, the model of Bickerman et al. reverses the terms, generating a melody for a given harmonic sequence (as is common practice in Jazz music), but both situations are clearly different to the case of autonomous melodic generation.

The model of Bickerman et al. uses a set of interconnected Restricted Boltzmann Machines (RBMs) to learn hierarchical structure from a training corpus of 4-bar Jazz “licks,” set to a cycling 4-chord progression: ii-V-I-VI⁷. RBMs are simple two-layer neural network models, with a single visible layer and a single hidden layer. A more detailed description of RBMs can be found in Bengio’s survey [16]. DBNs are built by associating the hidden layer neurons of one RBM with the visible layer neurons of another, resulting in multi-layered architectures in which the hidden layer representation at one level becomes the input feature vector at another. By interconnecting RBMs in this way, the DBN can learn to estimate

features of features, building complex hierarchical representations. Bickerman et al. used a 30-member bit vector to represent both the melodic and harmonic structure of the inputs to the DBN. Vectors were produced from the source music at a predetermined temporal resolution of 12 notes/vectors per beat, and data was passed to the DBN for training using a 4-bar sliding window, incremented one beat at a time. Prior to generation, the DBN was seeded with a sequence of vectors representing the desired harmonic progression and randomly assigned melody bits. The chord bits were clamped, so that they would not be altered by the stochastic nature of the RBMs. For melody generation, a similar sliding window approach was used, and melodic bits were generated by the system for each consecutive beat. After generating a beat, its melody bits were clamped in order to provide a stable melodic input representation as a context for the continued generation.

Bickerman et al. reported that the model was able to follow the chord progression accurately, while generating well-formed melodic patterns, and also that it performed similarly at different transpositions when trained on transposed inputs (i.e., the representation itself was not transpositionally invariant). Unfortunately, there is no detailed discussion of melodic structure, and the limited number of example generations provided make it difficult to determine whether higher-level melodic form was generated. On the other hand, it is also unclear to what degree higher-level melodic structure was demonstrated by the training set. The authors do note that a grammar-based melodic generator provided with the “Impro-Visor” Jazz education tool [114] (used during evaluation) generated subjectively superior output to the DBN model.

3.7 Integrated Cognitive Architectures

The design of Integrated Cognitive Architectures (ICAs) focuses on modelling intelligent behaviour at an architectural level. The field dates back to the early years of Artificial Intelligence (AI) research, and can generally be distinguished from the larger field of AI expert systems by its emphasis on modelling domain-general intelligence. Though studies investigating domain-specific performance of ICAs like ACT-R, SOAR, Icarus, SASE, Cerebus, and others, are common, the general goal of such architectures is to model intelligence without the requirement of task-specific knowledge and/or heuristics, which should be acquired by the agent through perception and action.

As Vernon points out in his overview of cognitive systems [235], the notion of cognition proposed by ICAs is fundamentally *embodied*, drawing on functions of perception, action,

deliberation, and motivation, and is generally opposed to more insular conceptions of cognition as a purely mental process of abstract reasoning. However, although all ICAs embrace this notion of embodiment as situated perception and action, there exist two fundamentally different views on the nature of an intelligent agent's relationship to its environment. According to Vernon, the *cognitivist* view proposes that the world has a concrete objective reality that is fundamentally logical, and can be accurately expressed through a symbol system. Thus, although cognitivist agents are situated, due to their positivist underpinnings they do not, in principle, need to be embodied. Their representational systems are assumed to provide a reliable interface with the world, and therefore interaction with the representation alone may suffice for generating intelligent behaviour [235]. The *emergent* view, on the other hand, makes no assumptions about the objective reality of the outside world, and does not assume that it can be expressed through a symbol system. Representations therefore are not assumed to have any correspondence with objects in the environment, and are only interpretable inasmuch as they give rise to behavioural responses in the agent. With no external symbol system to “interpret,” emergent agents must necessarily be embodied, since there is likewise no internal representational system to be operated upon—or at least no representational system with direct correspondences to items in the environment. Emergent cognitive systems are thus dynamic and self-organizing, and must generally be understood through their behaviour alone, since examination of their internal state may reveal little about their knowledge [235].

Another important aspect of cognition is robustness. A cognitive system must respond to unexpected conditions in an appropriate way, and should also learn to anticipate and avoid such situations in future. Clearly this implies some form of learning, but on a more subtle level it also suggests that learning should not be exclusively “declarative” (i.e., describing knowledge “about” things), but should also include implicit knowledge acquired through interaction with the environment (e.g., object avoidance and navigation), as well as procedural knowledge of how to achieve certain goals. Generally speaking, “explicit,” declarative knowledge captures an agent's intentional, goal-directed learning about a given subject or context. Implicit knowledge, on the other hand, consists of knowledge or skills acquired unintentionally while carrying out specific tasks [130]. ICAs tend to represent such explicit and implicit forms of knowledge separately, as in the Clarion architecture [222], which conceptualizes them on two different “levels.” Clarion is interesting in this regard, as the top level, which deals with explicit, declarative knowledge, is fundamentally cognitivist in nature. That is, it represents knowledge symbolically, thus ensuring that such knowledge

remains accessible to external analysis. The lower, implicit level, on the other hand, is implemented using a connectionist approach [84], and is thus fundamentally emergent and inaccessible to external analysis [222].

As implied by the term “integrated,” ICAs are defined by the interaction of a set of modules, each responsible for carrying out certain tasks, or handling certain capabilities of the agent. Most ICAs have modules for handling perception, planning, deliberation, and action, and also data structures representing different types of memory; short-term memory (STM), long-term memory (LTM), and procedural memory. The ACT-R architecture, for example, while lacking a specific short-term or working memory “module,” represents short-term, task-specific knowledge through a series of “buffers,” responsible for storing relevant data. Information is drawn from long-term memory into the short-term buffers through a process of “activation,” in which perceptual input is matched against representations stored in LTM [7, 142]. Any task-relevant knowledge found in LTM is moved into the short-term buffers, where it can be applied to performing specific tasks. Such tasks are treated as “goals” of the system, and are represented as declarative “chunks” in memory. Additional information required for the task, that was not found in LTM, is learned by the system and associated with its implicit knowledge of the perceptual/environmental context of the task, and with the explicit knowledge used in completing the task. Thus, when encountering the situation again in future, the system may draw directly on information previously applied to achieving the goal.

Unlike the ICAs introduced above, MusiCog does not attempt to model domain-general cognition. Rather, it uses the underlying theoretical approach of ICA design to integrate previous ideas from the fields of cognitive psychology, music perception and cognition, musical knowledge representation, and generative music, in the hope of gaining a better understanding of computational modelling of *human* music composition. The following chapter will present MusiCog in detail.

Chapter 4

MusiCog: An Integrated Architecture

We now introduce MusiCog, an integrated cognitive architecture for symbolic music learning and melodic generation. The development of MusiCog has been motivated by our desire to bring together research in music perception and cognition, musical knowledge representation, and generative music, for the creation of an autonomous musical agent. In its current state of development, MusiCog focuses only on the acquisition of musical knowledge that has been experimentally demonstrated to be accessible to musically untrained listeners; we do not explicitly model music theoretical or compositional knowledge. For this reason, we consider melodic generation in MusiCog to be “musically naïve”; i.e., drawing as much as possible on knowledge acquired through listening only, and not on any explicit representation or formalization of compositional thought. To the extent that familiar music informs our capacity to conceive of new music, MusiCog’s melodic output can be considered a form of *musical style imitation*. Although style imitation is a well established field [46, 51, 54, 73, 180], an adequate definition is conspicuously absent in the literature. We propose the following:

Given a style S , a style imitation system aims to generate pieces that would be classified as S by an unbiased observer.

Once again, however, it is important to note that we did not set out to create an optimal style imitation system with MusiCog. Rather, we sought to investigate the underlying hypothesis that, by integrating ideas from music psychology and the cognitive modelling of music, musical style imitation might arise as an implicit form of behaviour.

Just as the integrated cognitive architectures discussed in Section 3.7 are modular in design, so too is MusiCog divided into a set of processing modules, which work in conjunction when carrying out perceptual/cognitive tasks. An overview of the MusiCog design is

shown in Figure 4.1. Before entering into a detailed description of the various algorithms, we will introduce the main processing modules and outline the music descriptors used for input. MusiCog operates on MIDI information, and symbolic representations (Section 4.1) are assumed throughout this discussion.

MusiCog is fundamentally a symbolic cognitivist model. However, it should be noted that we do not maintain a cognitivist conception of cognition. Rather, we take a symbolic approach because it offers a relatively straightforward way of modelling concepts from the experimental literature in music perception and cognition, and also has the advantage of being easily accessible to analysis. This is in stark contrast to the approach taken in our previous work on the HSMM [161], for example, which had the disadvantages of high complexity and inaccessibility, making it extremely difficult to interpret the states of the system. Working with a symbolic system, we are able to trace the relationship between perception, learning, and action; in this case, the generation of musical output. Since our primary goal was to produce a musical agent amenable to analysis during both perception and generation, our symbolic cognitivist approach was pragmatic rather than theoretical.

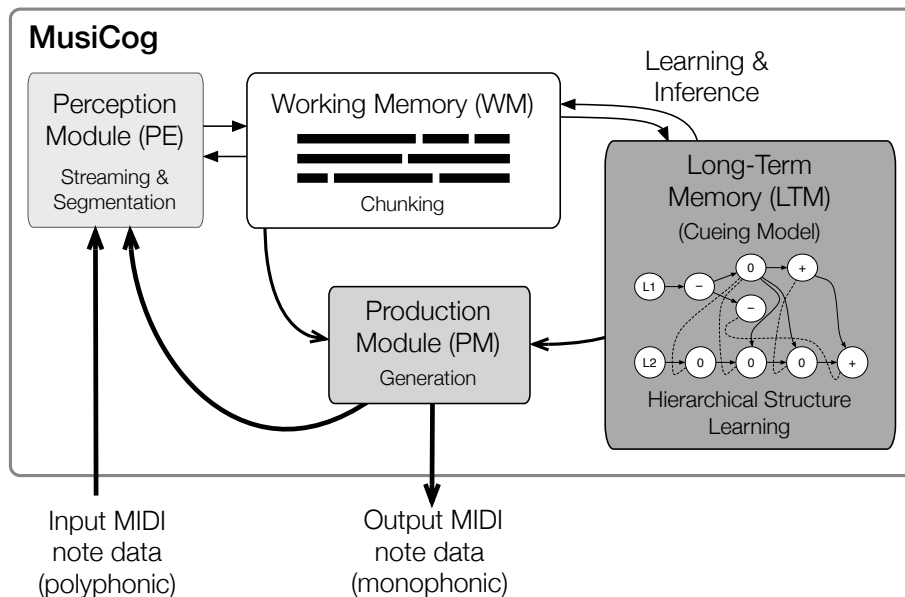


Figure 4.1: An overview of the MusiCog architecture.

The architecture of MusiCog is shown in Figure 4.1, and consists of a set of four processing modules:

- **Perception Module** (PE): Responsible for accepting musical input, separating polyphonic material into *streams* (Section 3.2), and performing low-level melodic segmentation (Section 3.1, p. 56) on each stream. In the PE, streams are considered analogous to the music theoretical notion of “voices” (i.e., independent monophonic parts¹).
- **Working Memory** (WM): A temporary memory for musical input, responsible for chunking familiar patterns (Section 3.4) and maintaining the set of active streams. Items are retained in WM as a function of the musical structure of the input in combination with several cognitive factors, as discussed in detail Section 4.2.2.
- **Long-Term Memory** (LTM): A Long-Term Memory representing the hierarchical structure of monophonic musical voices. The LTM is backed by our “Cueing Model” (CM) algorithm and data structure [160], which can learn from multiple, concurrent voices, and can create associations between the different voices in polyphonic textures, thus building an implicit representation of harmony. Formally, it is a mixed graph in which the directed edges represent sequential event transitions and the undirected edges represent associations between concurrent/synchronous events. During training the CM learns a set of directed subgraphs, each of which represents a different level of musical structure, similar to “time-span reductions” in the GTTM [146]. A detailed description of the CM is given in Section 4.2.3.
- **Production Module** (PM): Responsible for the generation of musical output. During generation the PM draws on the corpus-based knowledge of musical structure learned by the CM, and also on material held in WM, as described in Section 4.2.4. Output from the PM is directed back to the PE as input, forming a feedback loop, which facilitates self-evaluation and leads to complex, non-deterministic behaviour.

4.1 Music Descriptors

MusiCog takes either live MIDI performance data or standard MIDI files as input. At the time of writing there is no tempo/beat-tracking function, so events are assumed to be time-stamped on input with reference to an external beat. When processing MIDI files, the beat

¹Note that this is in contrast to Cambouropoulos’ model, as discussed in Section 3.2. The implementation of a variant of his approach is of interest for a future version of MusiCog (see Section 8.1).

implicit in the specification of MIDI ticks is used. As discussed in Section 4.2.3, input descriptors are divided into three main categories, corresponding to three degrees of musical specificity: Schema, Invariance², and Identity. Schema descriptors represent general qualitative changes that can be associated with a metaphorical sense of *direction*. A common example, used for melodic pitch sequences, is the notion of “melodic contour”, which is commonly described as ascending (+), descending (-), or repeating (0). The Invariance category employs descriptors that are more specific than the Schema category, but retain a degree of generality by remaining invariant under some specific transformation. For example, melodic *pitch intervals* are transpositionally invariant, just as beat-based timestamps are invariant to changes in tempo. Finally, the most specific category of Identity descriptors, represents the absolute, non-invariant values of events—e.g., MIDI note numbers, absolute timestamps (i.e., from the beginning of the work), Inter-Onset Intervals in ms (IOI—time between consecutive onsets), and so on.

MusiCog represents pitch Identity using the conventional descriptor of MIDI note numbers (C4 = 60 = middle-C). The pitch Invariance descriptor is the pitch interval (difference in semitones), and pitch contour (+, 0, -) is used for the Schema representation. Note inputs are timestamped using real-numbered beat values, such that whole numbers indicate even beats and fractional values indicate offsets within the beat. We use the IOI in ms to represent rhythmic Identity values. For rhythmic contour (Schema), we use the notation described above, corresponding to rhythmic lengthening (+), repetition (0), and shortening (-), and for rhythmic Invariance we use a novel descriptor called the “Beat Entry-Delay” (*bED*). The *bED* indicates the duration in beats of an onset from the local beat of the *preceding* onset. Figure 4.2 gives an example of the *bED* representation of a simple rhythmic pattern. In the figure, the local beat can be seen as a window, grouping together events with onsets occurring within the span of a single beat. Once an event has been processed within the current window, the beat associated with that window becomes the frame of reference for coming events, until a new beat arrives. When $bED = 1$, this indicates that the event falls directly on the beat³ (events 1, 5, and 6), whereas a fractional *bED* value indicates an offset position within the beat. For event 3, $bED = 1.5$ indicates that the event occurs halfway through the beat, but that the beat of the last onset occurred *one beat earlier*—i.e., the

²This term is used not in its computational sense, but rather in its musical sense, indicating values that are invariant under some specific transformation (e.g., the invariance of melodic interval content under pitch transposition).

³We use the value 1 to indicate on-beat events so that the *bED* value can be added directly to the previous (truncated) onset time; e.g., $[2.5] + 1.0 = 3.0$. An on-beat value of zero would prohibit such simple decoding.

referent event occurred in the previous window. Since event 3 establishes a new window, event 4 is then measured in relation to this new window, resulting in $bED = 0.75$.

Event	t_1	t_2	t_3	t_4	t_5	t_6
Timestamp	1	1.5	2.5	2.75	3	4
bED_t	1	0.5	1.5	0.75	1	1

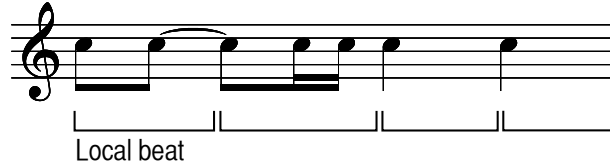


Figure 4.2: An example of the Beat Entry-Delay music descriptor.

Thus, for fractional values where $bED > 1$ (e.g., event 3) the integer part indicates the number of beats since the last detected beat/window. The function for deriving bED_t from two consecutive beat-based timestamps (n_{t-1}, n_t) is given in Equation 4.1. In a generative context, when given the previous timestamp n_{t-1} and bED_t , a new timestamp n_t can be calculated using Equation 4.2.

$$bED_t(n_{t-1}, n_t) = (n_t - \lfloor n_t \rfloor) + (\lfloor n_t \rfloor - \lfloor n_{t-1} \rfloor) \quad (4.1)$$

$$n_t(n_{t-1}, bED_t) = \begin{cases} \lfloor n_{t-1} \rfloor + bED_t + 1 & \text{if } bED_t \leq (n_{t-1} - \lfloor n_{t-1} \rfloor) \\ \lfloor n_{t-1} \rfloor + bED_t & \text{otherwise} \end{cases} \quad (4.2)$$

Since the bED representation encodes onset times with reference to a local beat, it is particularly convenient for generation, as it allows generated rhythms to automatically align with the beat structure of the new musical context. This can be seen in Figure 4.3, where the bED pattern (1.5, 0.75, 1, 1) from Figure 4.2 is rendered as a continuation of an altered rhythmic introduction (we have added a pitch contour for emphasis). It will be noticed that the recontextualized pattern derived from the bED retains its original relation to the beat, whereas the IOI-derived pattern (1, 0.25, 0.25, 1), encoded from the same original sequence and rendered as a continuation of the same rhythmic introduction, does not. Although the IOI-based rhythm retains the time interval ratios from the original sequence (i.e., in its internal structure), the continuation ignores the rhythmic shortening introduced by the altered context, resulting in a highly syncopated overall pattern—a pattern that has

actually been uniformly *shifted* one sixteenth-note back in time. In a generative context, such shifting can be awkward, since it is often carried over multiple measures.

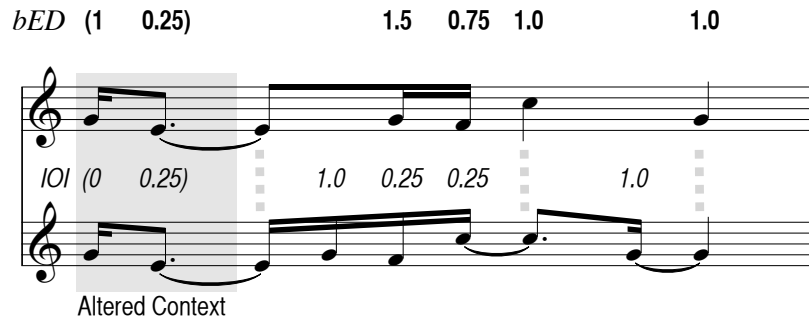


Figure 4.3: The *Beat ED* retains the relationship between rhythm and beat, whereas the *IOI* does not.

It is worth noting that the *bED* will also faithfully reproduce syncopated patterns in novel contexts, as shown in Figure 4.4⁴. Because pulse and metre are tightly bound to processes of rhythm classification in listeners [61], style-based generation that inadvertently shifts from straight rhythmic patterns to highly syncopated patterns (as in the *IOI* pattern in Figure 4.3), or vice versa (as in the *IOI* pattern in Figure 4.4), runs the risk of violating the rhythmic tendencies of the musical style being modelled. Because the *bED* represents relative durations in terms of an established beat, we believe it reflects the internal state of listeners more accurately than conventional *IOI*-based representations. To the best of our knowledge, this specific approach to rhythmic representation is unique in the literature.

Rhythmic Quantization

Because time is a continuous musical attribute and MusiCog is a discrete symbolic model, it is useful to limit the potential number of rhythmic representations using some form of *quantization*. In MusiCog, all rhythmic values are quantized using a fuzzy clustering method, in which each cluster represents a unique rhythmic value. The quantization function begins with a default set of clusters, representing common rhythmic values, as might be expressed

⁴Note that when deriving the new timestamp for the first generated *bED* in this example, case 1 from Equation 4.2 is used since $0.25 \leq (0.75 - 0)$

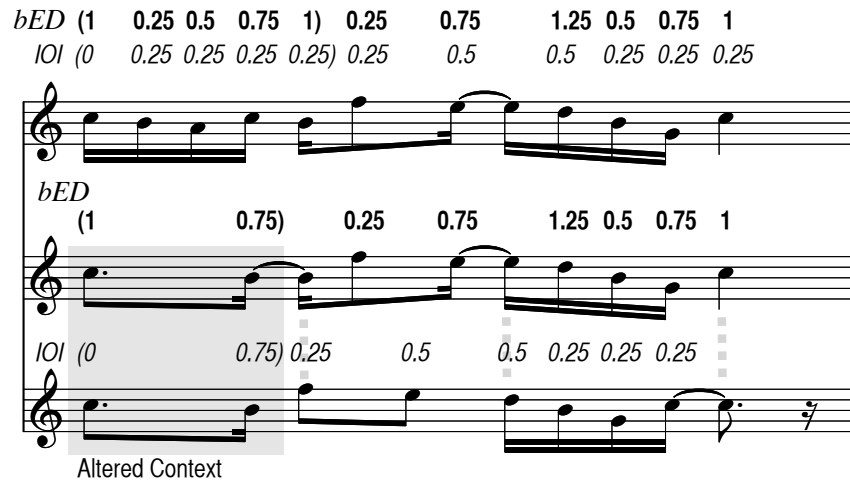


Figure 4.4: The bED also retains correct beat relationships in syncopated contexts.

by music notation⁵. When a rhythmic input is received, the quantization algorithm (Algorithm 4.1) iterates over the set of stored clusters C and calculates the (triangular) fuzzy membership R of input r , given each stored cluster value. We use a triangular window to indicate that, in rhythmic notation, there is a conceptually “ideal” rhythmic value, represented by the peak of the triangle. The cluster C_q that best classifies the input is identified as the index with the maximum membership in R (i.e., $\text{argmax}(R)$). If this membership exceeds a given threshold value λ , the input is quantized to the value of the stored cluster C_q , otherwise, a new cluster is created for the input value r . The RHYTHMICREPRESENTATION function gets the appropriate rhythmic value from element ε (i.e., so that the descriptor can be classified by the clusters stored in C). The fuzzy membership of r in C is calculated on line 5, where r is the input value, c is a stored cluster centre, and v is a parameter controlling the width of the membership window. Note that the membership function adjusts the width of the window proportionately to the value of r so that smaller rhythmic values receive a narrow window and larger rhythmic values a wider window. Rhythmic values that are not tempo invariant, like the IOI in milliseconds, are recalculated using the quantized values (e.g., the IOI in ms, if needed, is recalculated from the IOI in beats, given the current tempo).

⁵The rhythmic values used are: {1.0, 2.0, 0.5, 4.0, 0.25, 0.333334, 1.5, 0.75, 1.25, 0.666667, 1.333334, 1.666667, 3.0, 0.125, 0.2, 0.4, 0.6, 0.8, 0.375, 0.166667}. These values were chosen to provide relevant quantization points for both bED values and beat-based IOIs

Algorithm 4.1 Rhythmic quantization

```
1: function QUANTIZEINPUT( $\varepsilon, C$ ) // Quantize input  $\varepsilon$  in set of clusters  $C$ 
2:    $r \leftarrow$  RHYTHMICREPRESENTATION( $\varepsilon$ ) // Get rhythmic value from  $\varepsilon$ 
3:    $R \leftarrow \emptyset$ 
4:   for  $i \leftarrow 1, |C|$  do
5:      $R_i \leftarrow 1 - \left| \frac{r-c}{\delta} \right|$  // Fuzzy membership function
6:     if  $R_i < 0$  then  $R_i \leftarrow 0$ 
7:     end if
8:   end for
9:    $q \leftarrow \operatorname{argmax}(R)$ 
10:  if  $R_q > \lambda$  then
11:     $C_q \leftarrow C_q + \frac{r-C_q}{2}$  // Shift centroid slightly toward  $r$ 
12:     $r \leftarrow C_q$ 
13:  else
14:     $C_{|C|+1} \leftarrow r$  // Add  $r$  to the set of clusters
15:  end if
16:  return  $r$ 
17: end function
```

4.2 Processing Modules

We will now describe in detail the data structures and algorithms used for processing within each of MusiCog's main modules.

4.2.1 Perception Module (PE)

The PE is the first module in MusiCog's processing chain and is responsible for accepting musical input, separating polyphonic material into *streams*, and performing low-level melodic segmentation on each stream. It operates using three primary data structures:

- **Element** ε : The basic atomic type handled by MusiCog. *Elements* can be individual events (i.e., musical notes) or *chunks* (Section 4.2.2). Each unique element occupies a single unit of WM memory capacity. For convenience, we use dot notation to identify properties of individual musical events; e.g., MIDI note = ε .pitch, onset time = ε .onset.
- **Group** $G_t = \{\varepsilon_i : \varepsilon_i \in \Sigma\}$: The set of attacking or sustaining musical events at the current time step. The events in a group are unique elements from a finite alphabet Σ , containing the set of MIDI pitches and quantized rhythmic values.

- **Stream** $S = \{\varepsilon_k, \dots, \varepsilon_l\}$: An ordered collection of elements, stored in the WM. A stream is implemented as a FIFO structure, so that ε_k is the *oldest* element. The set of all currently active (non-empty) streams is notated as Ω_t . Each stream maintains its own state ${}^S\delta_t$, as described on page 99 below.

The functions of polyphonic voice-separation (or stream segregation) and melodic segmentation are closely related. Although voice-separation must logically be performed first, the same underlying notion of continuity or *cohesiveness* informs both processes. Generally speaking, voice-separation involves assigning each event in group G_t to a separate stream, such that the melodic cohesiveness of each stream is maximized. This process is often conceptualized in terms of voice-leading *cost*, where cost is inversely proportional to cohesiveness; i.e., low cohesiveness = high cost [36, 111, 113, 116, 156]. Melodic segmentation, on the other hand, occurs when changes in pitch or rhythm reduce this sense of cohesiveness, leading to the formation of perceptual boundaries. The task of the PE is thus to find the set of stream/event pairings with the lowest total cost, and to locate points of decreased cohesiveness in each stream and label these as segment boundaries.

At initialization, the PE contains an empty set of streams $\Omega_t = \emptyset$. When MIDI inputs are received, they are collected into a new group G_t such that the onsets of all events fall within the same temporal window (40 ms in our implementation). With no active streams, the PE creates a new stream for each event. For future inputs, when $|\Omega_t| > 0$, events in G_t are assigned to streams using a cost-based voice-separation algorithm similar to other gestalt-based approaches [36, 111, 156]. In general, such algorithms attempt to minimize the total voice-leading cost across all streams. The PE's cost calculation includes measures for 1) *pitch proximity*, 2) *rhythmic proximity*, 3) *melodic well-formedness*, 4) *predictability*, and 5) the avoidance of *voice-crossings*. We consider voice-leading cost in terms of the melodic "cohesiveness" caused by assigning a given input ε_t to a particular stream S , so that the cost of a particular voice assignment is proportional to the complement of its cohesion. The entire process of calculating the voice-leading cost for an input event $\varepsilon_t^{\text{cost}}$ and the melodic cohesion associated with that event $\Phi\varepsilon_t$, is given in Algorithm 4.2. We will describe the various steps in the algorithm with a focus on the music psychological principles underlying each calculation.

1) *Pitch proximity* ${}^P\rho$ is a $[0, 1]$ value indicating the distance between adjacent MIDI notes. It is calculated as a function of the melodic interval between consecutive pitches

P_{t-1} and P_t . The function used for the non-zero case (line 12) is designed to assign approximately equal values to step-wise motions, and to decrease more rapidly for leaps greater than a major 3rd.

2) *Rhythmic proximity* $R\rho$ (line 16) is also a $[0, 1]$ value based on a non-linear scaling of the IOI in seconds. The function is designed to fall to zero for IOI values greater than 5 seconds in duration, in order to approximate the influence of short-term memory retention times on low-level rhythmic grouping [216].

3) *Melodic well-formedness* is conceptualized in terms of expectancy, and combines pitch and rhythm factors. Pitch expectancy is already partially accounted for by the pitch proximity calculation, since stepwise motions tend to be more expected than leaps [216]. However, we also include a more specific pitch expectancy χP calculation, based on Huron's general observation that melodic patterns tend to gravitate toward the mean pitch of the melodic context. Huron's model [108] can be seen as a generalization of Narmour's notion of "post-skip reversal" [175]; the tendency for leaps of the "implicative interval" (i.e., the interval preceding the current interval) to be followed by reciprocal movement in the opposite direction. However, Huron's approach has the advantage of generalizing to cases where leaps are followed by continuations, as long as the continuation moves closer to the mean. We implement this form of expectancy in the PE's RETURNTO MEAN function (line 20), which takes the implicative interval I_{t-1} , the current interval I_t , and the running mean pitch \bar{P} as arguments. If interval I_t moves the input pitch closer to the mean pitch \bar{P} , and I_{t-1} is greater than 2 (i.e., larger than a major 2nd), the pitch expectancy χP is calculated as a function of the size of the implicative interval (otherwise $\chi P = 0$).

Algorithm 4.2 Calculate melodic voice-leading cost and cohesion

```
1: procedure MELODICCOHESION( $S, \varepsilon_t$ )           // Estimate cost and cohesiveness of
// extending stream  $S$  with event  $\varepsilon_t$ 
2:    $\varepsilon_{t-1} \leftarrow \text{GETMOSTRECENTEVENT}(S)$  // Get most recent event in stream
3:    $a \leftarrow 0.82$ 
4:    $b \leftarrow 0.18$ 
5:    $I \leftarrow \varepsilon_t.\text{pitch} - \varepsilon_{t-1}.\text{pitch}$  // Get melodic interval
6:    $\bar{P} \leftarrow \text{UPDATEMEANPITCH}(S, \varepsilon_t)$  // Calculate running mean pitch
7:    $\text{IOI} \leftarrow \varepsilon_t.\text{onset} - \varepsilon_{t-1}.\text{onset}$  // Get IOI in seconds
8:    $\text{OOI} \leftarrow \varepsilon_t.\text{onset} - \varepsilon_{t-1}.\text{offset}$  // Get OOI in seconds
9:   if  $I = 0$  then
10:  |  ${}^\rho P \leftarrow 1$ 
11:  else
12:  |  ${}^\rho P \leftarrow 1 - \frac{1}{1+6e^{-\frac{x}{2}+2}}$  // Pitch proximity calculation
13:  end if
14:   ${}^\rho R \leftarrow 0$ 
15:  if  $0 < \text{IOI} < 5$  then
16:  |  ${}^\rho R \leftarrow 1 - \frac{1}{1+\text{IOI}^{-\frac{e}{2}}}$  // Rhythmic proximity calculation
17:  else
18:  |  ${}^\rho R \leftarrow 0$ 
19:  end if
20:   ${}^\chi P \leftarrow \text{RETURNTOMEAN}(I_{t-1}, I_t, \bar{P})$  // Pitch expectancy function
21:   ${}^\chi R \leftarrow \text{DESAINEXPECTANCY}(\text{IOI}_{t-1}, \text{IOI}_t)$  // Rhythmic expectancy function
22:   $\Phi P \leftarrow f_{\max}({}^\rho P, {}^\chi P)$ 
23:   $\Phi R \leftarrow f_{\max}({}^\rho R, {}^\chi R)$  // Combine proximity and ex-
// pectancy to estimate cohesion
24:   $\Phi PR \leftarrow f_{\min}(\Phi P, \Phi R)$  // Combine pitch and rhythm cohesion
25:   $v \leftarrow 1$ 
26:  if  $0 < \text{OOI} < 2$  then
27:  |  $v \leftarrow 1 - 0.77 \times \log_{10}(\text{OOI})$  // Calculate loss of cohesion over rest
28:  else
29:  |  $v \leftarrow 0$  // Rests  $\geq 2$  s eliminate cohesion
30:  end if
31:   $\Phi PR' \leftarrow \Phi PR \times v$  // Scale cohesion for duration of rest
32:   $\varepsilon_t^{\text{cost}} \leftarrow 1 - \Phi PR'$  // Calculate base cost
33:   $\varepsilon_t^{\text{cost}} \leftarrow a(\varepsilon_t^{\text{cost}}) + b(1 - \varepsilon_t^{\text{pred}})$  // Include predictability cost
34:   $x \leftarrow \text{VOICECROSSINGCOST}(\Omega, \varepsilon_t)$  // Calculate voice-crossing cost
35:   $\varepsilon_t^{\text{cost}} \leftarrow f_{\min}(\varepsilon_t^{\text{cost}}, x)$  // Final cost includes voice-crossing cost
36:   $\Phi \varepsilon_t \leftarrow f_{\max}(1 - \varepsilon_t^{\text{cost}}, \varepsilon_t^{\text{pred}})$  // Event cohesion combines complement
// of final cost with predictability
37: end procedure
```

For the rhythmic aspect of melodic well-formedness, the DESAINEXPECTANCY function (line 21) calculates the rhythmic expectancy χR using the “basic expectancy” measure from Desain’s “(De)composable Theory of Rhythm” [60]. Desain’s theory models the general observation that listeners tend to expect rhythmic continuations that form low-integer ratio relationships with the most recent IOI. His “basic expectancy” calculation estimates the expectancy of the current IOI ratio r_t as a sum of gaussians centred around a series of low-integer ratios $\{\frac{1}{n}, \dots, \frac{1}{2}, 1, 2, \dots, n\}$ [60] ($n = 7$ in our implementation). The adaptation used in the PE is shown in Equation 4.3, where χR is the estimated expectancy, and j is a control on the width of the gaussians ($j = 5$ in our implementation). The function is plotted in Figure 4.5.

$$r_t = \frac{IOI_t}{IOI_{t-1}}$$

$$\chi R = \frac{\sum_{k=1}^7 \frac{1}{k} e^{(jk - j\frac{1}{r_t})^2} + \sum_{k=1}^7 \frac{1}{k} e^{(jk - jr_t)^2}}{2} \quad (4.3)$$

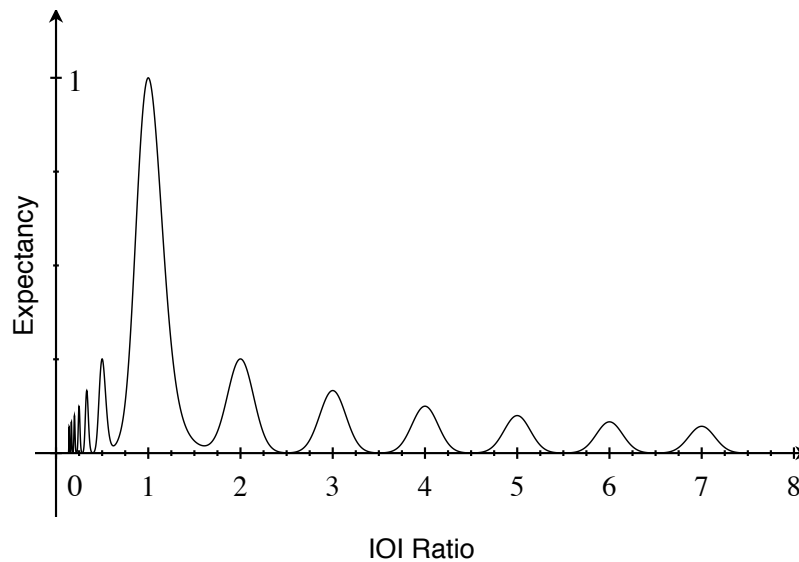


Figure 4.5: The rhythmic expectancy function, based on Desain’s “basic expectancy” from his “(De)composable Theory of Rhythm.”

It is difficult to quantify the interaction of proximity and expectancy values as they impact on the perception of melodic cohesiveness. In some cases proximity simply obviates expectancy, as can happen in rapid melodic passages, which tend to be grouped primarily

by proximity. In other cases, however, sudden changes of IOI can promote the perception of melodic boundaries even in the presence of relatively close proximity relations. For this reason, we use the combination function shown in Equation 4.4 when combining proximity and expectancy values into a composite “cohesion” value. This function uses the value of x to scale the value of y , reducing the influence of y in the combination.

$$\begin{aligned}
 & a, b \in \mathbb{R} \\
 & x = \begin{cases} \max(a, b), & f_{\max} \\ \min(a, b), & f_{\min} \end{cases} \\
 & y = \begin{cases} \min(a, b), & f_{\max} \\ \max(a, b), & f_{\min} \end{cases} \\
 & \zeta(x) = 1 - \frac{1}{1 + e^{-x\pi^2 + \frac{\pi^2}{2}}} \\
 & f(x, y, \zeta(x)) = \frac{x + \zeta(x)y}{1 + \zeta(x)}
 \end{aligned} \tag{4.4}$$

When applying Equation 4.4 the bias can be weighted toward either the greater or lesser of a pair of inputs a and b by assigning x and y to either $\max(a, b)$ or $\min(a, b)$ accordingly. Thus, in order to reduce the influence of low values on a weighted combination, the assignment $x = \max(a, b)$, $y = \min(a, b)$ can be used, so that low values of either a or b have less impact on the final combination. Conversely, the assignment can be reversed, so that lower values have a greater influence on the final combination. We will refer to these two versions of the combination function as $f_{\max}(a, b)$ and $f_{\min}(a, b)$.

Because the pitch expectancy calculation tends to return low values, except in the case of “post-skip reversals,” we use the $f_{\max}(a, b)$ function when combining pitch proximity and expectancy values on line 22 of Algorithm 4.2. Dynamically weighting toward the higher value helps suppress the influence of frequently low expectancy values. Since proximity also tends to have a strong influence on rhythmic grouping, we again use the $f_{\max}(a, b)$ function when combining rhythmic proximity and expectancy on line 23. The resulting pitch and rhythm cohesion values, ${}^\Phi P$ and ${}^\Phi R$, serve two purposes: 1) to estimate the perceptual cohesiveness of assigning a given element ε_i to a particular stream S (i.e., for use during voice-separation), and 2) to identify transitions in a given stream that might indicate perceptual boundaries (i.e., for melodic segmentation purposes).

In a manner similar to the interaction of proximity and expectancy, the way in which pitch and rhythm factors jointly influence the cohesiveness of melodic sequences is also complex

and highly context sensitive. In many situations rhythm appears to have a stronger influence [115], while in other cases pitch factors alone should be considered (e.g., in melodic passages with uniform rhythm). To model the context sensitivity of this relationship, we combine pitch and rhythm factors (line 24) using the $f_{\min}(a, b)$ version of the combination function. This is intended to model a simple form of attention; i.e., since high cohesion implies continuity, low cohesion implies discontinuity, change, or surprise, which should be attended to. The use of $f_{\min}(a, b)$ allows variations in the lower of the two cohesion values to exert a stronger influence on the overall cohesion, thus increasing the PE’s sensitivity to local decreases of cohesiveness in either pitch or rhythm.

Once the pitch and rhythm cohesion values have been combined, we apply a uniform scaling v in order to model the presence of rests. The amount of scaling is proportional to the duration of the rest, as shown on line 27, where OOI is the Offset-Onset Interval (i.e., the time between the previous offset and the next onset) in seconds. In our implementation we scale a logarithmic curve by 0.77 so that scaling will reach zero by the end of a 2-second rest, modelling the strong tendency for silences to demarcate perceptual boundaries. Having estimated the cohesion of a transition, we then calculate the base cost $\varepsilon_t^{\text{cost}}$ as the complement of the cohesion, so that high cohesion equals low cost, as shown on line 32.

4) In order to account for cognitive “top-down” factors based on familiarity [205], we also include a *predictability* measure $\varepsilon_t^{\text{pred}}$, based on the contents of WM. As outlined in Section 4.2.3, the WM provides a $[0, 1]$ estimate of the predictability of a given input, based on a pattern-matching search over the chunks and segments currently held in WM. The predictability can be used to bias the cost of voice-leading choices toward patterns that have been observed previously in the musical context. The base cost is updated to account for predictability on line 33, as a weighted combination using weights a and b . The weights in our implementation ($a = 0.82$, $b = 0.18$) were determined through experimentation and may have different optimal values for different corpora.

5) In polyphonic textures, *voice-crossings* occur whenever two or more voices have intersecting pitch registers, such that the lower voice transitions to a higher pitch than the higher voice. Such crossings jeopardize the integrity of independent melodic lines [32], and are generally avoided in voice-leading and counterpoint. To model this tendency, the VOICECROSSINGCOST function (line 34) calculates the voice-crossing cost as the number of crossings a particular voice-leading assignment (S, ε_t) would incur, divided by the total number of streams $|\Omega_t| - 1$. Since avoidance of voice-crossings is generally an important

principle of voice-leading, the base cost and the voice-crossing cost are combined using the $f_{\min}(a, b)$ function so that increases in voice-crossing cost will have a significant influence on the base cost; this determines our final voice-leading cost. The last line of Algorithm 4.2 updates the cohesion of event ε_t given stream S as a combination of the complement of the final cost and the predictability. The $f_{\max}(a, b)$ version of the combination function is used to prevent low predictability values—a “top-down” factor in MusiCog—from providing too much influence over the “bottom-up” gestalt principles that are the focus of the PE.

When streams outnumber the events in group G_t (i.e., $|\Omega_t| > |G_t|$), voices are assigned iteratively, starting with the lowest-cost stream/event pairing (S, ε_t) and removing pairings after each assignment. However, when $|\Omega_t| \leq |G_t|$, the streams must compete for events. In this case, we use an heuristic based on the notion of “compromise” to order the iterative assignment process. For each pairing (S, ε_t) , the compromise is the cost difference between the two lowest-cost solutions—i.e., the cost *penalty* that would be incurred by taking the *second-best* solution. The best overall solution can be considered the one requiring the least overall compromise. Therefore, we reverse-sort the streams according to their calculated compromise values, then iterate over pairings, assigning the lowest-cost pairing on each iteration (i.e., so that the lowest-cost voice-leading choice is given to the stream with the highest potential compromise), and removing pairings after each assignment.

Once the events in G_t have been assigned to streams, the PE attempts to locate low-level segment boundaries for each active stream. Because we are modelling online perception, we cannot rely on the use of a look-ahead function, as is common in melodic segmentation algorithms (for an overview, see [184]). However, after several attempts with more complex approaches, we found that by combining the cohesion ${}^\Phi\varepsilon_t$ with the predictability $\varepsilon_t^{\text{pred}}$ (line 36), an acceptable segmentation could be achieved. This approach is based on the notion of maintaining cohesion between the stream’s most recent segment S_1 and event ε_t . The segmentation algorithm thus identifies as a boundary β any event that causes an instantaneous decrease in cohesion. This is shown in Equation 4.5, where σ_Φ is the running standard deviation of cohesion, and φ is a scaling factor referred to as the “cohesion tolerance,” which permits small decreases in cohesion to occur before identifying a boundary. In order to avoid one-note, singleton segments, we also include a routine to group singletons with either the preceding or following segment, based on minimizing local differences in cohesion.

$$\beta_t = \begin{cases} 1, & \text{if } {}^\Phi\varepsilon_t - (\sigma_\Phi \varphi) < {}^\Phi\varepsilon_{t-1} \\ 0, & \text{otherwise} \end{cases} \quad (4.5)$$

When PE processing is complete, the incoming events from group G_t will be separated into $n > 0$ streams, and bottom-up, event-level segment boundaries will be defined. It is worth noting that, using the dynamic combination function in Equation 4.4, we were able to remove several parameters from our original PE design [160].

Mode and Tonality Induction

There is one final perceptual phenomenon modelled in the PE, added primarily for use in the melodic generation process. In order to maximize flexibility and context sensitivity during generation, the PM utilizes melodic interval information when determining pitch output. A potential problem with this approach is that generated intervals may not conform to the scale structure of the current musical context, leading to undesirable chromatic alterations in the output. In order to mitigate the effects of this problem, it is useful to include some form of pitch quantization, so that generated pitches remain in the desired scale/mode. Of course, in an online system designed to learn musical structure in a style-agnostic manner, we cannot arbitrarily choose a pitch scale in advance. Rather, we would prefer that the system respond to the scale and tonality arising in perception (if, indeed, such a perception arises). For this reason, rather than pre-selecting a target scale as a parameter in MusiCog, we chose to include a form of scale/mode induction. There are two music psychological principles underlying our approach: 1) that human listeners tend to more easily induce scales that are constructed from sequences of unequal intervals [229], and 2) that in many musical styles, frequency of note usage, note duration, and accent structure tend to emphasize certain pitches over others.

To model this general notion of the scale as a perceptually organized pitch space, we include two steps; one to estimate the mode, and another to estimate the tonal centre. The result of these estimations is expressed through a “confidence” rating that indicates the degree to which a single mode and tonality can be induced from the given musical context. In MusiCog, we refer to the “tonal centre,” as opposed to the “tonic,” to emphasize the fact that we are not modelling tonality in the music theoretical sense of “functional harmony,” but are interested only in finding the most salient pitch in a given context. Similarly, our interest in “modes” lies not in a bias toward modal (or even diatonic) composition, but rather in a desire to acknowledge the apparent cognitive bias toward pitch scales of 5 to 7 notes, formed by sequences of unequal intervals [72, 218]. Our implementation was inspired by Huron and Parncutt’s tonality induction method [106], which models the influence of echoic memory (Section 3.1) by allowing the spectral content of *sequential* events to overlap in

time. Using this representation, Huron and Parncutt were able to calculate the mutual harmonic support of non-simultaneous pitches in melodic contexts, improving upon the tonality induction results of previous systems that lacked the echoic memory function [106]. Unlike Huron and Parncutt, who worked with audio content, we are working with symbolic representations and therefore do not have access to spectral information. For this reason, we chose to synthesize a hypothetical spectral energy distribution for each MIDI note using the set of harmonic weights shown in Table 4.1. The intervals in this table represent the spectral components being synthesized (as defined by their interval distance from the “fundamental” MIDI note) and the weights represent their relative synthesized amplitudes. By using this scale, and calculating the summed spectral content of all melodic pitches in a given stream, we are able to mimic the tonality induction behaviour of Huron and Parncutt’s model.

Harmonic	MIDI Interval	Weight
1	0	3.16
2	12	1.0
3	19	0.7
4	24	0.52
5	28	0.39
6	31	0.3
7	34	0.22
8	36	0.15
9	38	0.09

Table 4.1: Synthesized spectral weights used for calculating the harmonic support of adjacent and/or simultaneous MIDI pitches.

In our implementation, we build a chroma vector by summing the synthesized spectral content for all elements in the WM stream, where the contribution of each pitch-class is proportional to the element’s duration. The induced tonality is decided as the peak index of the chroma vector (i.e., $\text{argmax}(\text{chroma})$). Once the tonality is estimated, we rotate the stream’s chroma vector to the pitch-class index of the induced tonality and perform mode induction by multiplying the rotated chroma vector by a binary “masking” vector. The masking vector is based on the Ionian mode and multiplication is carried out across each of its 7 diatonic rotations. The sum of the cross-product is used as an estimate of the induction function’s support for the given mode. The mode with the greatest support is considered the best mode for the melodic passage. This process is demonstrated in Table

4.2, using a hypothetical chroma vector, the weights of which suggest a tonal centre on G in the minor/Aeolian mode.

	Ionian	Dorian	Phrygian	Lydian	Mixolydian	Aeolian	Locrian						
Mask rotations	0	2	4	5	7	9	11						
	C	C#	D	E \flat	E	F	F#	G	A \flat	A	B \flat	B	SUM
	0	1	2	3	4	5	6	7	8	9	10	11	
Chroma	0.3	0	0.5	0.2	0	0.3	0.1	0.8	0	0.2	0.4	0	
argmax =	7												
Ionian Mask	1	0	1	0	1	1	0	1	0	1	0	1	
Chroma rotation (7)	0.8	0	0.2	0.4	0	0.3	0	0.5	0.2	0	0.3	0.1	
Mask rotation 0	1	0	1	0	1	1	0	1	0	1	0	1	
Chroma x Mask =	0.8	0	0.2	0	0	0.3	0	0.5	0	0	0	0.1	1.9
Mask rotation 2	1	0	1	1	0	1	0	1	0	1	1	0	
=	0.8	0	0.2	0.4	0	0.3	0	0.5	0	0	0.3	0	2.5
...													
Mask rot. (Aeolian) 9	1	0	1	1	0	1	0	1	1	0	1	0	
=	0.8	0	0.2	0.4	0	0.3	0	0.5	0.2	0	0.3	0	2.7
Mask rotation 11	1	1	0	1	0	1	1	0	1	0	1	0	
=	0.8	0	0	0.4	0	0.3	0	0	0.2	0	0.3	0	2

Max sum of (Chroma x Mask) = 2.7 = Aeolian Mode (minor)

Table 4.2: Estimating the mode using the chroma vector and a modal “masking” vector.

Since induction of 7-note scales is not appropriate in all musical situations, we include a *confidence* measure, which can be used to determine whether pitch quantization should be applied. Here, confidence is proportional to the standard deviation of non-zero support ratings across all modes. The underlying assumption is that highly chromatic passages will receive support for several modes, and that therefore the standard deviation across all non-zero estimates should be relatively low. Conversely, music that fits well in a particular mode will be highly supported for that mode, and receive relatively low support for the others, resulting in a greater standard deviation across all non-zero estimates. Using the confidence, we can determine whether or not pitch content should be adjusted to better fit a given mode, so that highly chromatic material can be left unquantized when confidence is low. During training, mode/tonality induction is run on all streams in WM, and the confidence rating updated with each pass of the induction function. Through this process we

are able to derive a mean confidence level for each training corpus, which can be used as a reference during generation.

4.2.2 Working Memory (WM)

As mentioned in the introduction to this chapter (p. 74), the WM provides temporary storage and maintains the set of active streams. It also exploits two new data structures, *segments* and *chunks*, to model the cognitive phenomenon of “chunking” (see Section 3.4):

- **Segment** $v^{L_n} = (\varepsilon_1, \dots, \varepsilon_k)$: A finite sequence of elements, where L_n indicates the formal level of the segment (i.e., the amount of hierarchical nesting)⁶. We index the individual elements in a segment using subscripts: $v_1^{L_n}, \dots, v_k^{L_n}$. Segments are constructed in the PE using gestalt-based segmentation rules (see Section 4.2.2 above), and may contain discrete musical events (L_1) or chunks (L_2, L_3, \dots, L_n). The initial note event in a segment has a special perceptual/cognitive significance and is referred to as the “boundary event” βv^{L_n} . When a segment is learned or inferred by the LTM, it is assigned a “terminal node” $v^{L_n} \theta_k^i$, which represents a reference to the segment in LTM. This process is outlined in Section 4.2.3.
- **Chunk** $C(v^{L_n})$: A chunk is essentially a “wrapper” around a segment. As such, chunks demonstrate the same principles of hierarchical nesting as segments. The important difference between chunks and segments is that a chunk represents a segment that has been committed to LTM and subsequently recognized. Further, chunks can be grouped together into higher-level segments, which can themselves also be “chunked” (i.e., via recognition in LTM). Whereas segments are treated by the WM as sequences of elements occupying $|v^{L_n}|$ spaces in memory, chunks are treated as single elements, regardless of the amount of nesting. Because chunks always contain events at the lowest hierarchical level, every chunk will have a boundary event $\beta C(v^{L_n})$. Likewise, every chunk will also have a terminal node $C(v^{L_n}) \theta_k^i$, taken from the wrapped segment during the chunking process (see Section 4.2.2).

The WM builds streams as FIFO structures of elements, which may be segments or chunks. Elements are retained in WM based on a combination of factors including 1) element *predictability*, 2) element *recency*, 3) musical *parallelisms* [31, 146] between the element and all other elements stored in WM, 4) *habituation* to the element, given the WM

⁶For compactness, the level notation can be simplified to v^1, v^2, \dots, v^n .

contents [155], and 5) the basic storage *capacity* of WM. These factors are combined in a measure of “cognitive salience,” which provides an estimate of the element’s utility for understanding the larger musical context [252]. The complete process of calculating the cognitive salience is given in Algorithm 4.3. It is important to note that we do not increase the cognitive salience⁷ of elements. Rather, we impose a scaling on the salience, adjusting its rate of decay according to a number of cognitive factors. The remainder of this section will outline in detail the relevant operations in Algorithm 4.3.

Algorithm 4.3 Update cognitive salience of elements in WM

```

1: procedure UPDATECOGNITIVESALIENCE( $\Omega$ )
2:    $\kappa \leftarrow 9$  // Capacity threshold
3:    $\psi \leftarrow 0.2$  // Saliency threshold
4:    $t \leftarrow 15$  // Outer limit on WM decay function
5:    $h \leftarrow 12$  // Time parameter for habituation function
6:   for  $S \in \Omega$  do
7:      $v^1 \leftarrow S_1$  // Get the most recent segment
8:      $v^{1sal} \leftarrow v^{1pred}$  // Initial saliency = predictability in LTM
9:      $V \leftarrow \text{FLATTENSTREAM}(S)$  // Flatten all elements in stream to
// a sequence of  $L_1$  segments
10:    for  $\varepsilon \in V$  do
11:      if  $|\varepsilon| > |v^1|$  then
12:         $\varepsilon \leftarrow \text{GETSUBSEGMENT}(\varepsilon, |v^1|)$  // Get subsegment of length  $|v^1|$ 
13:      end if
14:       $\tau \leftarrow v_k^1.\text{timestamp} - \varepsilon_k.\text{timestamp}$  // Calculate age of segment  $\varepsilon$ 
15:       $D \leftarrow 1 - \frac{1}{t} \frac{\tau}{1 - \psi}$  // Linear WM decay rate
16:      if  $D < 0$  then
17:         $D \leftarrow 0$ 
18:      end if
19:       $\Gamma \leftarrow 1$  // Set parallelism to 1
20:      if  $\varepsilon \theta_k^n \neq S_1 \theta_k^n$  then // If terminal nodes do not match
21:         $\Gamma \leftarrow 1 - \frac{1}{\kappa}$ 
22:      end if
23:       $\varepsilon^{sal} \leftarrow \varepsilon^{sal} \times D \times \Gamma$  // Scale saliency by recency and parallelism
24:       $H \leftarrow \varepsilon^{sal} \times \left( 1 - \frac{1 - (1 - \varepsilon_k^{sal})}{1 + e^{\tau + 12}} \right)$  // Habituation function
25:       $\varepsilon^{sal} \leftarrow \varepsilon^{sal} \times H$  // Scale saliency by habituation
26:    end for
27:  end for
28: end procedure

```

⁷It is also important to note that we do not currently model *perceptual* saliency; i.e., the relative importance of unexpected or novel events [8].

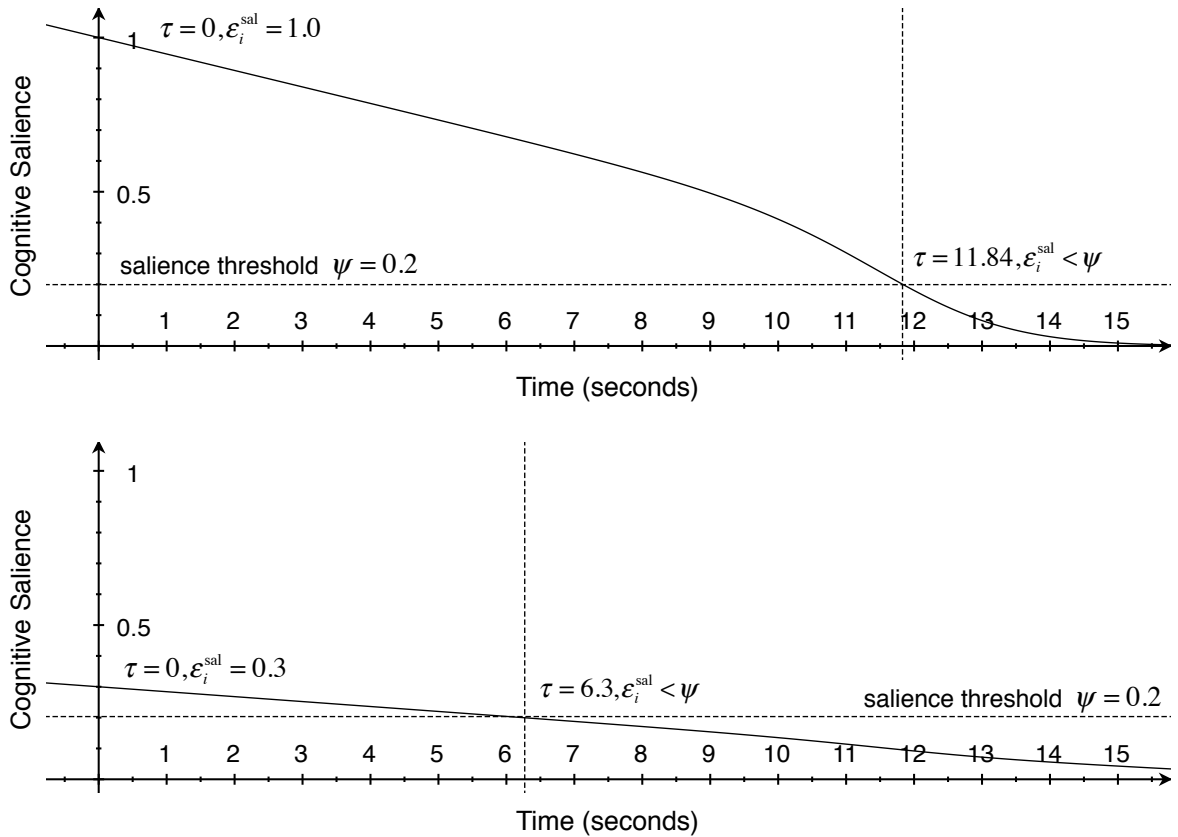


Figure 4.6: The decay of an element’s cognitive salience in WM as a function of its initial salience ϵ^{sal} , the duration for which it has been retained in WM τ , and habituation.

Before stream processing begins, the basic capacity threshold κ is set 9 to approximate the STM “rule of thumb” limit of 7 ± 2 elements⁸ [216]. The salience threshold ψ , outer duration limit t , and habituation time h have been determined ($\psi = 0.2$, $t = 15$ s, $h = 12$ s) so that low-salience elements will fall below the salience threshold after approximately 6 seconds, and high-salience items that *lack* strong support from musical parallelisms will fall below the threshold after approximately 12 seconds⁹, as can be seen in Figure 4.6.

1) The *predictability* of a perceived element ϵ^{pred} reflects the LTM’s capacity to predict the element in the current musical context, as discussed in Section 4.2.3. The predictability of a segment or chunk is the mean predictability of all its constituent elements. As indicated

⁸This is by no means a hard-limit in MusiCog, as will become clear during the following discussion.

⁹Without the influence of habituation such elements would remain in WM for up to 20 seconds. However, this is a purely hypothetical situation, since the scaling imposed by the habituation function is proportional to salience.

on line 8, we define an element's initial salience (i.e., at stream index 1) according to its predictability. Since all elements are initially added at stream index 1, this value is implicitly assigned to all elements in WM.

2) The *recency* (or “age”) of an element indicates the duration for which it has been retained in WM. For each segment/chunk, recency is calculated using the timestamp of its most recent element (ε_k on line 14). With no other influences, the salience of an element will decay as a linear function of its retention time, as modelled by the scaling value D calculated on line 15.

3) The contents of WM (and LTM) are used to calculate the degree of musical *parallelism* [31, 146] a given element shares with the current contents of the WM; i.e., how many musically similar elements are currently being stored. The estimation of musical similarity is a complex problem [11, 27, 31, 58], which admittedly deserves closer attention, but as a provisional definition we say that two segments (or chunks) are similar if they share the same Schema representation (e.g., matching pitch contour). The set of all such similar elements is thus analogous to Deliège's notion of the “imprint” [59]. If a given element lacks parallelisms with the current contents of WM, the rate of decay is accelerated (see line 21), reducing the element's longevity in WM. The parallelism is determined by comparing the terminal nodes of the two elements (line 20), such that if they are equal—specifically, they have matching Schema representations (discussed in more detail in Section 4.2.3)—the two elements are considered to express a parallelism.

4) Another important cognitive phenomenon influencing the longevity of elements in WM is *habituation*; the decline in response to a repeated stimulus [155]. To model habituation, we use the function on line 24 to scale the cognitive salience proportionally to its own value, and to its recency, so that elements that have maintained high cognitive salience for relatively long periods of time will have their salience more aggressively suppressed. The function introduces habituation-based scaling after approximately 10 seconds of retention in WM. When the number of stored elements exceeds the WM's capacity κ , any element with a recency greater than T seconds and a cognitive salience less than the salience threshold ψ will be discarded from WM. This process is carried out in the UPDATESTREAM function, given in Algorithm 4.4. Of course, salience is only one factor influencing retention, since chunking processes (discussed in the following section) can also support the retention of elements by embedding them in higher-level representations. Thus, WM retention time in MusiCog is a function of the musical content, so that there is no *a priori* maximum,

resulting in a highly dynamic memory model, in which the structure of the musical content is a primary factor in the model's retention capacity.

Algorithm 4.4 Update contents of stream

```

1: procedure UPDATESTREAM( $S, \varepsilon_t$ )           // Update contents of stream  $S$  given input  $\varepsilon_t$ 
2:    $\kappa \leftarrow 9$                            // Basic capacity = 9 elements
3:    $T \leftarrow 5$                                // Basic duration in WM = 5 seconds
4:    $\psi \leftarrow 0.2$                            // Cognitive salience threshold = 0.2
5:   if  $|S| > \kappa$  then
6:      $e \leftarrow S_k$                              //  $e$  is oldest element in stream
7:      $\tau \leftarrow \varepsilon_t.\text{timestamp} - e_k.\text{timestamp}$  // Calculate age of  $e$ 
8:     if  $\tau > T \wedge e^{\text{sal}} < \psi$  then
9:       REMOVEELEMENT( $e, S$ )                       // Remove  $e$  from stream
10:    end if
11:  end if
12: end procedure

```

Chunking in the WM

When a new stream S is created, an empty segment is added to the end of the stream. As the PE assigns events to streams, each new event is appended to the end of the segment at S_1 . When the PE detects a low-level segment boundary, a new segment is created for the boundary event, and inserted at S_1 . The WM then “closes” the segment at S_2 , indicating that it is complete. At each time-step, the WM passes the segment at S_1 (i.e., the most recent segment) to the LTM for learning. After learning/inference, the LTM assigns the segment a “terminal node” representing the learned state of the segment in LTM. With repeated exposure, a given segment’s representation in LTM becomes increasingly specific (see Section 4.2.3 for details). Segments that can be inferred by the LTM are said to be “recognized,” so that segments inferred at higher tiers are more familiar than those inferred at lower tiers (i.e., since each tier represents a degree of recognition¹⁰). When a segment can be perfectly recognized by the LTM—i.e., having all of its transitions inferred at the Identity tier—it is “wrapped” into a new chunk, which replaces the segment in WM. For each segment v^{Ln} that is converted to a chunk $C(v^{Ln})$ in this manner, WM capacity is increased by $|v^{Ln}| - 1$. The chunking process is illustrated in Figure 4.7.

¹⁰This notion of recognition is analogous to the idea of predictability, but is more specific with regard to past events. While the notes in a phrase we’ve just listened to are, by definition, perfectly predictable, this does not mean that the phrase was familiar or recognized.

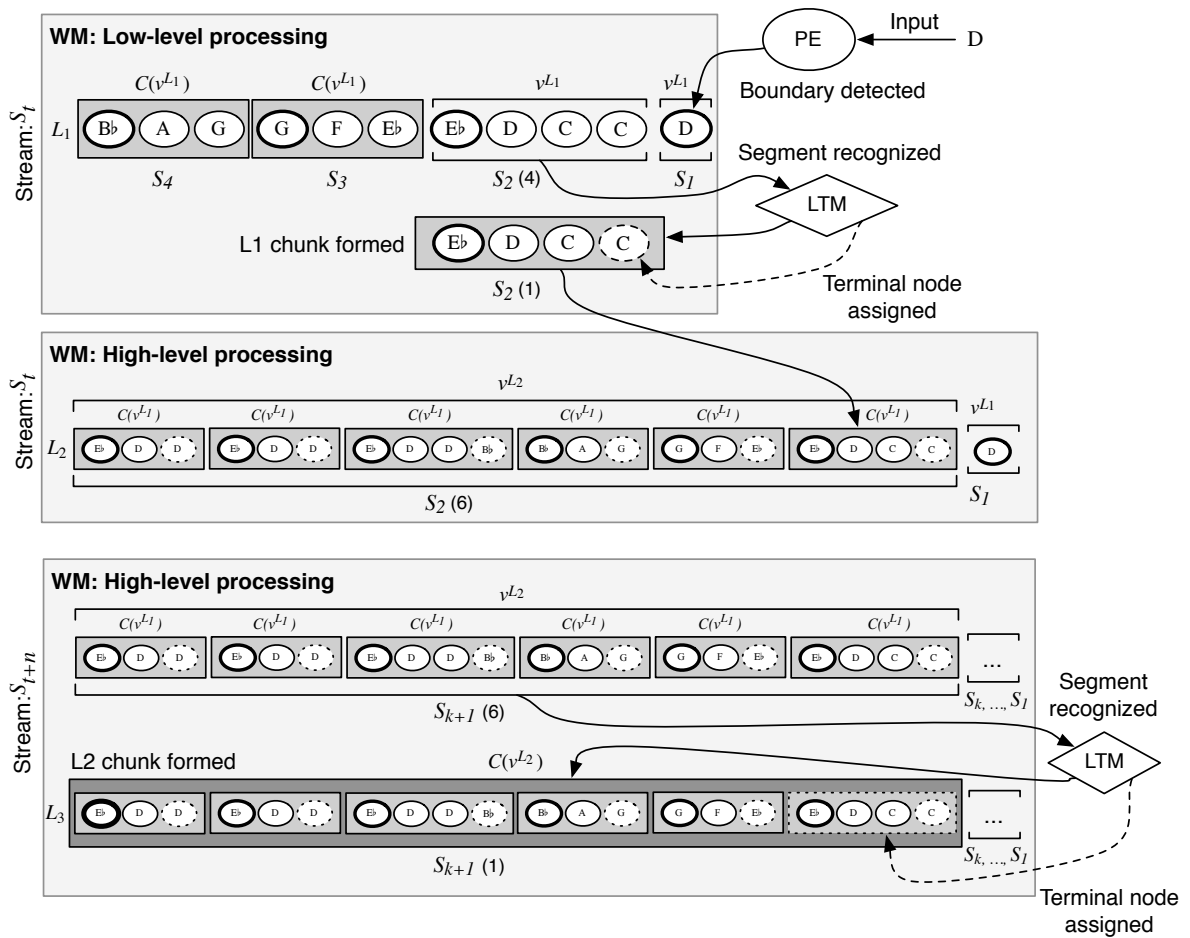


Figure 4.7: Chunking in the WM is based on the recognition of segments in the LTM.

Phrase Boundary Detection in the WM

The WM is also responsible for higher-level *phrase* segmentation, which is generally thought to be a top-down, cognitive process [31]. In MusiCog, phrase boundaries are associated with the detection of non-contiguous musical parallelisms between segments or chunks. The repetitions must be non-contiguous because contiguous repetitions would fail to convey a specific higher-level parallelism; or more precisely, they would represent some arbitrary set of potential parallelisms. This is illustrated in Figure 4.8, where the sequence of identical segments, 4.8a, can be partitioned arbitrarily into several higher-level chunks. Sequence 4.8b on the other hand, which contains a non-consecutive repetition,

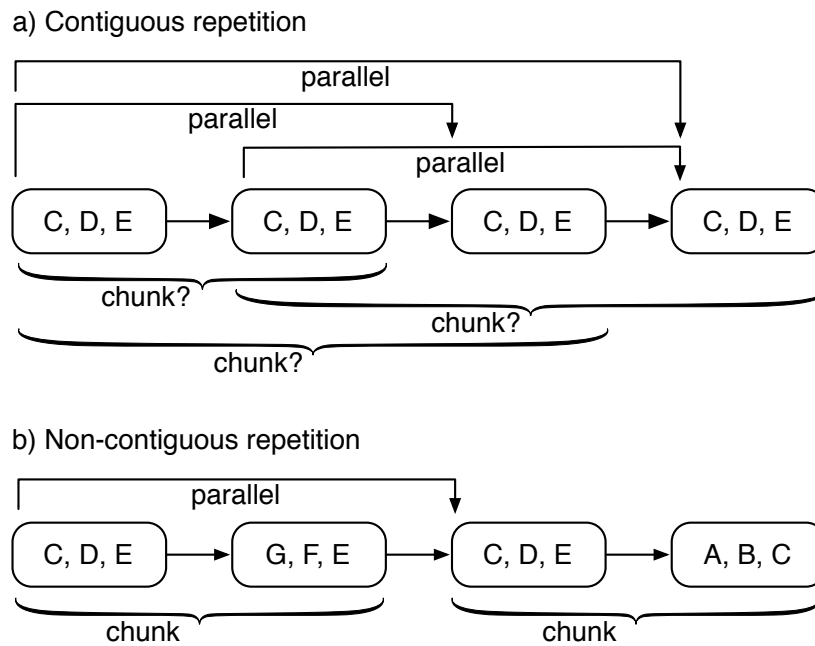


Figure 4.8: Higher-level chunks are formed by non-contiguous repetitions.

has a clear parallel structure that can be partitioned into only two chunks. It is the *reiteration of a motive introduced earlier* (and retained in WM) that signals the phrase boundary.

Looking at the melody from Mozart's 40th Symphony (Figure 4.9), we can see two distinct phrases, the second of which begins one scale-step below the first. The asterisk marks the point at which listeners will typically identify the phrase boundary. The WM finds these parallelism-based boundaries by looking for similarities between a newly formed chunk $C(v^{L_n})$ and the sequence of chunks already stored in WM. When a previously recognized chunk is detected in a larger phrase structure, this chunk can serve as a phrase boundary. The process of finding parallelisms in this manner is given in Algorithm 4.5.

The CHUNKPARELLELISM algorithm takes a stream S and a chunk $C(v^{L_n})$ as arguments. The algorithm iterates over the contents of S , starting with the oldest element S_k , and searches for a parallelism between the input chunk $C(v^{L_n})$ and a preceding chunk ϵ . If a parallelism is found, a new segment is started, to which all contiguous, intervening chunks are added, until chunk $C(v^{L_n})$ is reached, or the formal level of ϵ changes. The ISCHUNK function identifies whether the element is a chunk, and the GETLEVEL function returns the level of the chunk. The purpose of comparing the levels is to ensure that the chunk $C(v^{L_n})$ being tested as a boundary is at the same formal level as the chunk ϵ against which it is being tested. The ADDELEMENT function adds element ϵ to the end of segment v' , and the

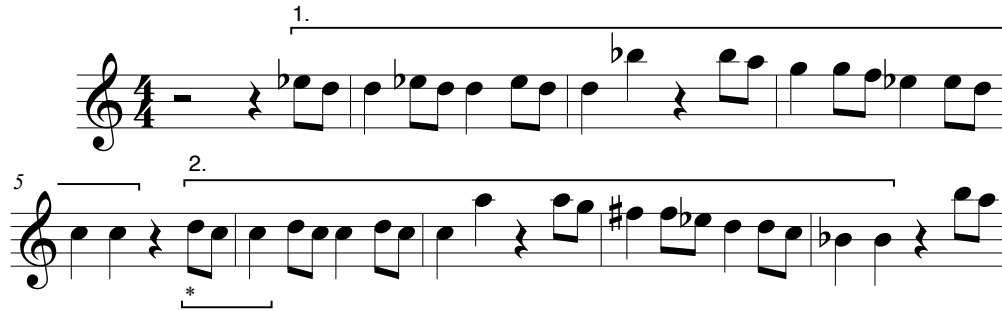


Figure 4.9: Parallelism in Mozart’s 40th Symphony. The asterisk indicates the point at which the musical parallelism will indicate a phrase boundary.

comparison $\varepsilon\theta_k^0 = C(v^L)\theta_k^0$ compares the terminal nodes of the input chunk and ε . Since the terminal node acts as a reference to the segment in LTM, two segments/chunks that share the same terminal node also share the same contour (for a detailed discussion, see Section 4.2.3). Because contour provides the grounds for the estimation of similarity, the comparison is made between Schema representations (tier 0) of ε and the input chunk. The boolean flag B is used to determine whether a new segment is currently being built, in which case it should be extended (otherwise, a new segment v' is started). The function returns \perp (i.e., null) if the generated segment v' is less than two elements long, since a single-element segment would fail to encode any higher-level structure¹¹. Otherwise, it returns the new segment v' ; a higher-level “segment-of-chunks,” the first chunk of which is similar to the input $C(v^{Ln})$.

¹¹For example, encoding the sequence (x,y,z) as a series of single-element sequences (x)(y)(z), does not provide any new information about the sequence. On the other hand, encoding the sequence (x,i,y,i,z,i,j) as (x,i)(y,i)(z,i,j) reveals the boundary relationships between x, y, and z, established by identifying the repetition pattern (n,i).

Algorithm 4.5 Parallelism-based chunking of elements in a stream

```
1: procedure CHUNKPARELLELISM( $S, C(v^{Ln})$ )           // Find parallelism for chunk  $C(v^{Ln})$ 
                                     // in stream  $S$ 
2:    $i \leftarrow \text{INDEXOF}(C(v^{Ln}), S)$            // Get index of  $C(v^{Ln})$  in  $S$ 
3:    $B \leftarrow \text{false}$ 
4:    $k \leftarrow |S|$ 
5:   while  $k > i$  do
6:      $\varepsilon \leftarrow S_k$ 
7:     if  $\text{ISCHUNK}(\varepsilon) \wedge \text{GETLEVEL}(\varepsilon) = n$  then //  $\varepsilon$  is a chunk at same level as  $C(v^{Ln})$ 
8:       if  $B = \text{true}$  then  $\text{ADDELEMENT}(\varepsilon, v')$            // Add element  $\varepsilon$  to segment  $v'$ 
9:       else
10:        if  $\varepsilon \theta_k^0 = C(v^{Ln}) \theta_k^0$  then           // Compare Schema terminals
11:           $v' \leftarrow \text{NEWSEGMENT}$            // Create a new segment
12:           $\text{ADDELEMENT}(\varepsilon, v')$            // Add  $\varepsilon$  to new segment
13:           $B \leftarrow \text{true}$ 
14:        end if
15:      end if
16:      else if  $B = \text{true}$  then break
17:      end if
18:       $k \leftarrow k - 1$ 
19:    end while
20:    if  $|v'| < 2$  then
21:      return  $\perp$ 
22:    else
23:      return  $v'$ 
24:    end if
25: end procedure
```

4.2.3 Long-Term Memory (LTM)

The primary function of the LTM is to create a persistent, generalized, long-term representation of the contents of the WM. To do this, the LTM uses a novel data structure and learning algorithm called the “Cueing Model” (CM). The CM is a development of our earlier “Closure-based Cueing Model” (CbCM) [162], and is a mixed graph representing the transitions within, and relationships between, segments and chunks from the WM. It is used to learn the contents of WM streams, and to make inferences on those contents. The CM is an online learner, which learns to approximate the formal structure of a corpus of training works. The model utilizes a number of new data types:

- **Model** M_n : The CM model, where n is the number of levels (defined below) in the model. Each level is represented as a directed subgraph corresponding to a level of

hierarchical form, as encoded by chunking processes in the WM, described in Section 4.2.2.

- **Level** L_n : A directed subgraph in the CM representing a level of formal structure, where n is the level number. L_1 represents sequences of events, L_2 represents sequences of chunks $C(v^{L_1})$, L_3 represents sequences of L_2 chunks $C(v^{L_2})$, and so on. The relationship between segments, chunks, and levels can be seen in Figure 4.7.
- **Node** $\eta_k^i(v)$: A node on a given level of the CM, where k is the depth (i.e., distance from the root) of the node, i is the “tier” of the node, and v is the “value.” The depth corresponds to the sequential position of an element in a learned segment. The notion of tiers is taken from our earlier work on the CbCM [162] and is described in more detail below (p. 100). A node in a trained model stores a music descriptor, learned at some tier of specificity (e.g., contour, interval, pitch), in a particular musical context. Nodes in a level are connected by weighted edges, which represent transitions between elements. Edges are only added to nodes of the same tier—e.g., $\tau(\eta_k^2(v), \eta_{k+1}^2(v))$ —and their weightings indicate the frequency of the transition. When identifying nodes in relation to their levels, we concatenate the level and node notations, so that $L_1\eta_3^1(7)$ identifies a “Level 1 Invariance node at depth 3, with a value of 7.” We also identify the *terminal node* as the last node visited by learning/inference of a given segment, notated as $v^{L_n}\theta_k^i$ (or for chunks, $C(v^{L_n})\theta_k^i$).
- **State** $\delta_t = \{\eta_k^i : \eta_k^i \in M_n\}$: The set of nodes returned by the CM learning algorithm (see p. 105 below) at time step t . The state may be discussed with reference to a single level $L_n\delta_t$, or to the set of states across all levels, resulting from inference of a given stream $^S\delta_t = \{^{L_1}\delta_t, \dots, ^{L_n}\delta_t\}$. For polyphonic music it is also possible to refer to the state of all streams $^\Omega\delta_t$, so that $L_n\delta_t \subset ^S\delta_t \subset ^\Omega\delta_t$.
- **Link** $\iota(a, b)$: A weighted, directed edge between nodes on adjacent levels. By connecting segments at different hierarchical levels, links encode the formal relationships between fundamental musical concepts like *motives* and *phrases*.
- **Association** $\alpha(a, b)$: A weighted, undirected edge between any two nodes in the CM. Associations are made when two nodes a and b are visited, or *activated*, at the same time step—i.e., when $(a, b) \in ^\Omega\delta_t$. The strength of the weighting represents the frequency of co-activation. When more than two nodes are simultaneously activated, pair-wise associations are created for all activated nodes.

Data Representation

Unlike the CbCM, which encoded surface events at all levels, the CM only encodes surface events at L_1 and L_2 . L_1 encodes relatively short sequences of events comparable to musical “motifs”—as dictated by PE segmentation—while L_2 encodes sequences of *boundary events* (which are “linked” to L_1 segments), to form an abstract representation of musical “phrases,” as illustrated in Figure 4.10. All higher levels encode sequences of paths from their adjacent *sub-levels*, so that L_3 encodes sequences of L_2 paths, L_4 encodes sequences of L_3 paths, and so on. In this sense, higher level nodes represent “background” structures, similar to those expressed by non-terminal symbols in a formal grammar [147].

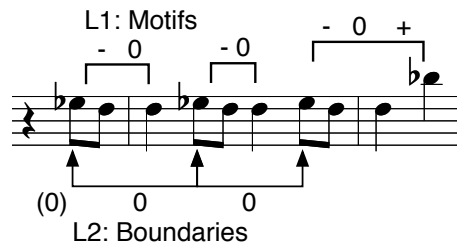
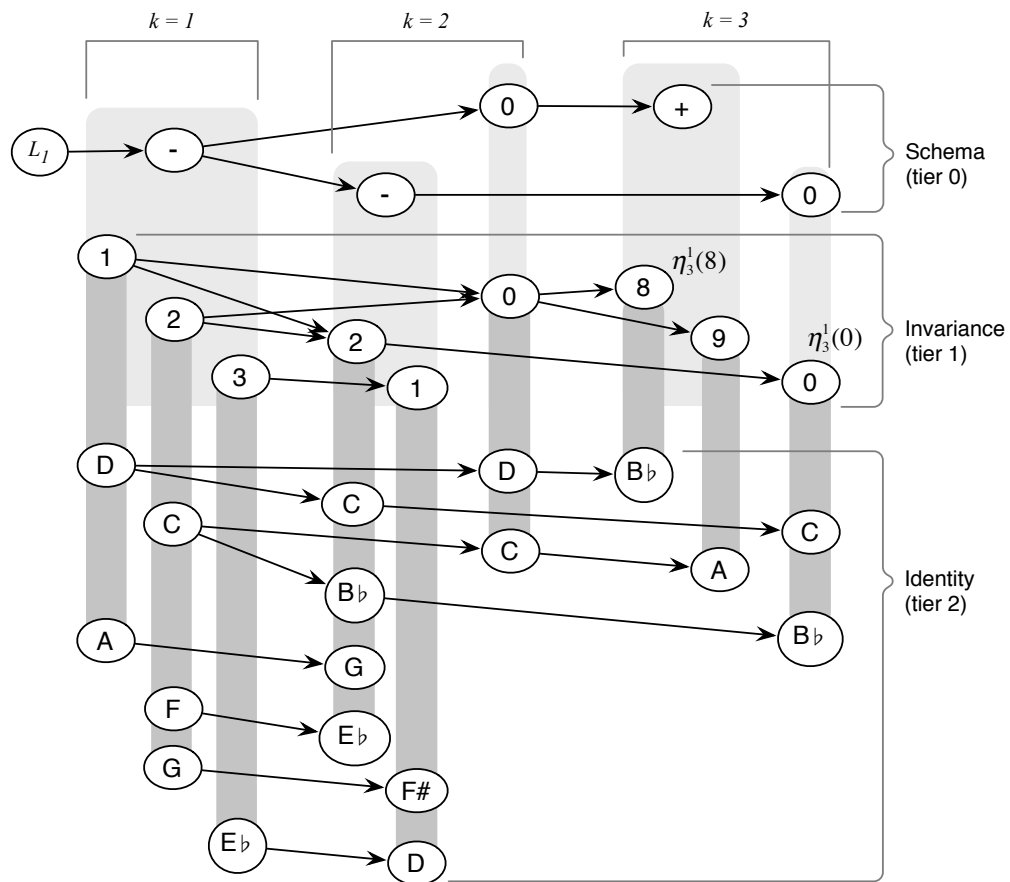


Figure 4.10: Encoding phrases as a combination of Motifs (L_1) and Boundaries (L_2).

The CM encodes events at different *tiers* of musical specificity (previously called “states” in the CbCM—see [162]): Schema (tier 0), Invariance (tier 1), and Identity (tier 2). The tier indicates the relative specificity of the node’s stored representation. It is useful to think of lower tier nodes “containing” higher tier nodes so that, for example, a tier 0 node could represent a pitch contour, and would contain a tier 1 node representing a pitch interval. This pitch interval node would in turn contain a tier 2 node representing the MIDI pitch itself. Figure 4.11 shows all three tiers of the L_1 pitch graph created by the two opening phrases from Mozart’s 40th Symphony. In the diagram, we see that each lower-tier node contains a set of higher-tier representations, and that looking from left to right, each tier forms a directed subgraph, at a different level of specificity. At tier 0 we see a pitch contour graph (Schema), which subsumes a pitch interval graph at tier 1 (Invariance), and a pitch graph at tier 2 (Identity). It is worth noting that the Schema graph will always be a tree, whereas the higher-tier graphs may exploit multiple parents, allowing for increased compression.

This also enables a form of classification, similar to that suggested by Ockelford in his “zygonic” theory [178], in which new structures are understood by comparison to previously learned structures. For example, if we look at the musical context implied by depth 3



Input

Phrase 1: (E_b, D, D) (E_b, D, D) (E_b, D, D, B_b) (B_b, A, G) (G, F, E_b) (E_b, D, C, C)

Phrase 2: (D, C, C) (D, C, C) (D, C, C, A) ($A, G, F\#$) ($F\#, E_b, D$) (D, C, B_b, B_b)

Figure 4.11: L_1 of a CM trained on the opening of Mozart's 40th Symphony, showing all three tiers.

Invariance nodes $\eta_3^1(8)$ and $\eta_3^1(0)$, we see a form of classification, illustrated in Figure 4.12. It is clear that both nodes provide terminals for closely related segments, and that although the segment marked with an asterisk is not in the original melody, it is nevertheless a viable melodic structure. It is also worth note that the 2-2-0 structure of $\eta_3^1(0)$ is actually encoded by the model *before it appears at the end of phrase 2*; i.e., the model has learned enough partial structure to predict the new structure before it occurs.

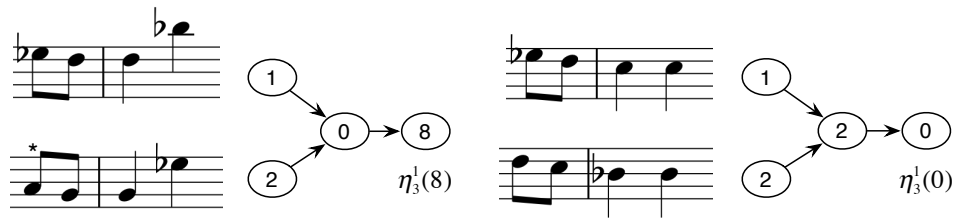


Figure 4.12: Automatic classification of segments at tier 1. Nodes $\eta_3^1(8)$ and $\eta_3^1(0)$ act as terminals for pairs of closely related segments.

The CM produces fundamentally the same hierarchical structure as the CbCM, but the connectivity between levels reveals significant differences. In our previous approach, inspired by Dubnov and Assayag’s *Incremental-Parsing* method [73], we attempted to learn the entire structure incrementally. This meant that boundary connections encoded the relationships between lower-level segment *endings* (i.e., “terminal nodes”) and higher-level segment boundaries. However, with the addition of the WM, the CM is able to learn directly from sequences of chunks, and can thus encode boundary relationships somewhat independently of the event-level sequential structure. It is worth noting that such a learning pattern suggests a *cognitive* (as opposed to a strictly *perceptual*) process, since chunking must be carried out before the higher-level structure can be learned. The difference between the CM and CbCM approaches is illustrated in Figure 4.13.

Here we see that, due to the incremental learning pattern, the CbCM encodes “+1” transitions at each L_2 node, representing the relationship between the D that ends each segment and the subsequent boundary E_b . In contrast, L_2 of the CM encodes only the relationships between boundaries (i.e., interval 0). Thus, although the L_2 Identity information is the same (i.e., sequences of E_b s), the encoded Schema and Invariance relationships are not. We believe the CM’s encoding to be a vast improvement, as it more accurately captures phrase-level structure and is thus better able to generalize across different musical situations. For example, a CM trained on the first two complete phrases of the Mozart

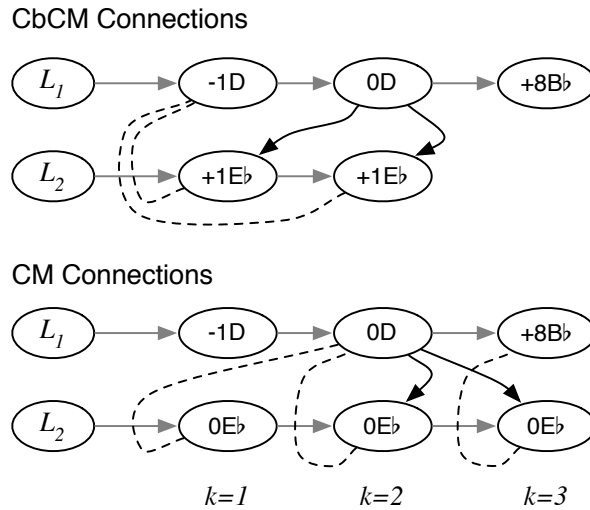


Figure 4.13: Different approaches to encoding form in the CbCM and the CM.

melody will infer the inherent similarity of *all* phrases built from the repetition of L_1 segments (motives), whereas the CbCM will not. Figure 4.14 provides an example of how a CM trained on Mozart's 40th can recognize, without any further training, the inherent similarity of a novel melody sharing the same series of contour relationships at L_2 .

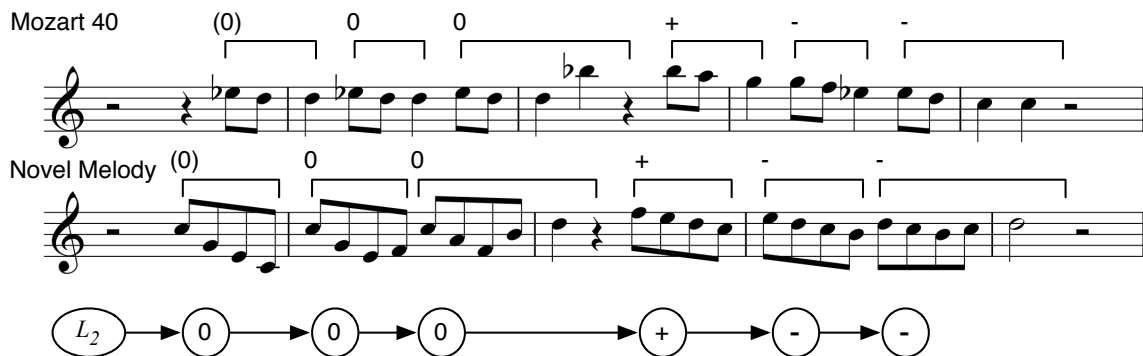


Figure 4.14: The second, novel melody can be inferred by the same L_2 schema structure as the Mozart melody.

Figure 4.15 shows the links connecting nodes on different levels. The dotted lines indicate *terminal links*, and the arrows between levels indicate *cueing links*. Terminal links are used to connect segment boundaries to lower-level terminal nodes (i.e., segment endings), whereas cueing links connect terminal nodes to *subsequent* boundaries. With regard to such linking connections, another important difference between the CM and the CbCM that can be seen by looking again at Figure 4.13, is that terminal links in the CM connect to

terminal nodes, whereas the CbCM always connects into depth $k = 1$; a consequence of incremental learning. This can be a serious disadvantage during generation, since depth 1 will generally be a high-entropy search space, reducing the likelihood of accurately recalling the L_1 segment originally associated with a learned L_2 boundary. The CM is able to make these more specific terminal links because it learns from complete chunks (which possess both a boundary node and terminal node) at higher levels, as facilitated by the WM. The strictly incremental learning process of the CbCM could not learn these connections, since the subsequent terminal could not be known at the time the boundary was established. From a cognitive modelling perspective, this emphasis on the connection between boundaries and terminals in the CM also reflects the tendency for listeners to more accurately recognize events near phrase boundaries [211]. Finally, it is worth noting that, whereas the CbCM always omits the first event (e.g., the initial E_b), the CM encodes *all* events on the musical surface.

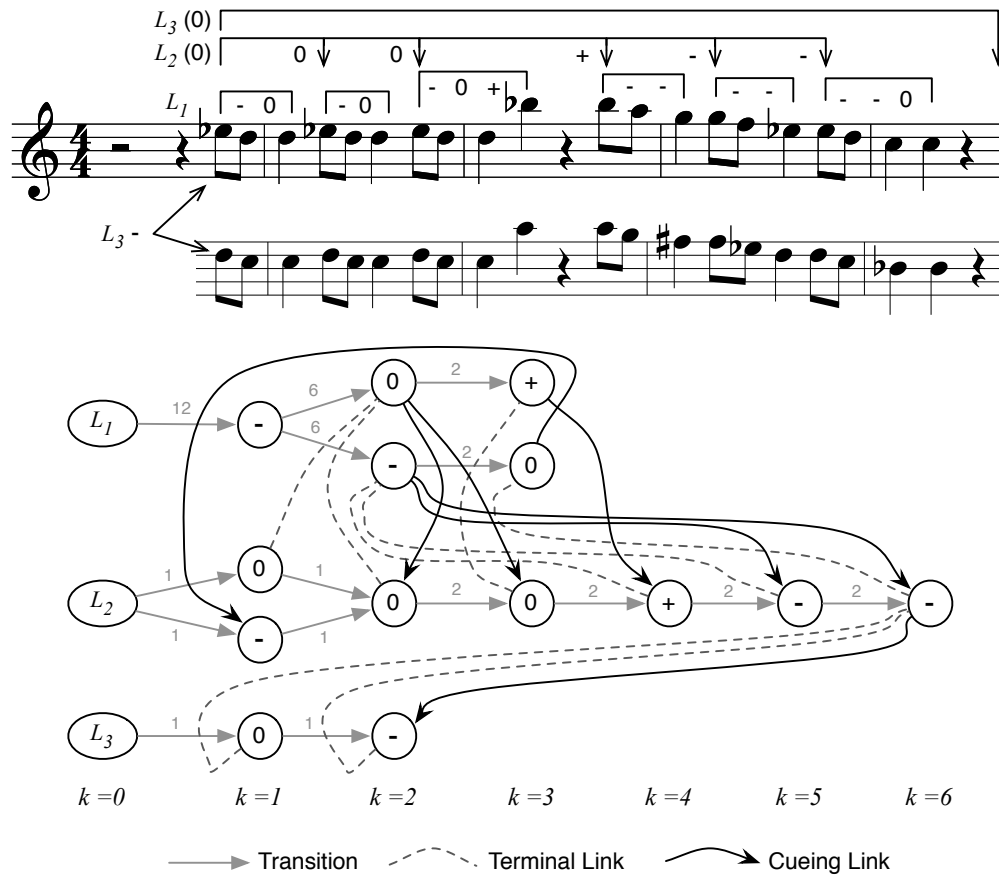


Figure 4.15: Schema view of a trained CM showing *transitions* (grey arrows), *terminal links* (dotted) and *cueing links* (black arrows).

Looking at Figure 4.16 we see a detailed view of the links connecting nodes on different levels. This example demonstrates the cueing of two consecutive segments (0 - 0 +), accessing nodes [1] to [4], and (+ - -), accessing nodes [5] to [7]. Although transitions between L_2 boundaries are encoded by edges at L_2 , the cueing links allow them to be inferred in cases where the L_2 context is not available, and also provide additional support for selecting transitions when multiple edges are present, as outlined in Section 4.2.4 (Figure 4.19), in our discussion of PM generation.

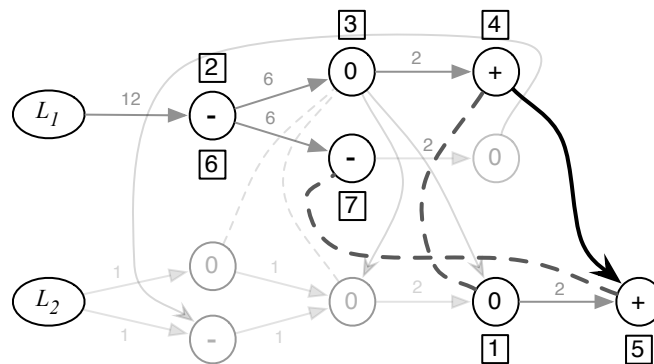


Figure 4.16: Detail view showing the *cueing* of sequence ((0 - 0 +) (+ - -)).

Learning and Inference in the CM

Learning in MusiCog involves adding nodes to the CM graph, such that the directed subgraph at L_1 encodes the series of transitions between the events in WM segments, and higher-level subgraphs encode the series of boundary relationships between WM chunks. As higher-level subgraphs are learned, so too are the termination and cueing relationships between nodes on different levels. Nodes for each transition are added at the highest tier possible, given any prior learning, so that the CM graph representation becomes increasingly specific with increased exposure to the WM contents. Schema tier transitions are added first, building schematic, contour-based representations. As transitions are received that can be predicted at the Schema tier, the CM automatically shifts to the Invariance tier, and begins learning invariant representations. Once the invariant transitions can be predicted, learning shifts to the final Identity tier.

Algorithm 4.6 The main WM update function

```
1: procedure UPDATEWM( $\Omega_t$ ) // Update all streams in  $\Omega$ 
2:   for  $S \in \Omega_t$  do
3:      $i \leftarrow |S|$ 
4:      $v' \leftarrow S_1$  // Get current segment
5:     if GETLEVEL( $S_2$ ) = 1 then
6:        $\bar{v}^1 \leftarrow S_2$  //  $\bar{v}^1$  is the penultimate segment in  $S$ 
7:     end if
8:     UPDATESALIENCE( $S, \bar{v}^1$ )
9:      $A \leftarrow$  CONTIGUOUSCHUNKS( $S$ ) // Get series of contiguous chunks in  $S$ 
10:    for  $k \leftarrow i, 1$  do
11:       $v' \leftarrow$  CHUNKPARALLELISM( $S, S_k$ ) // Parallelism-based chunking
12:      if  $v' \neq \perp$  then
13:        | PROCESSCONTIGCHUNKS( $v'$ ) // Train on high-level sequence  $v'$ 
14:        |  $C \leftarrow$  BUILDCHUNK( $v'$ ) // Create new chunk from  $v'$ 
15:        |  $S_k \leftarrow C$  // Replace  $S_k$  with new chunk
16:      else
17:        | PROCESSCONTIGCHUNKS( $A$ )
18:      end if
19:       $k \leftarrow k - 1$ 
20:    end for
21:    if  $\bar{v}^{1\text{pred}} = 1$  then // CM can predict segment
22:      |  $C \leftarrow$  BUILDCHUNK( $\bar{v}^1$ ) // Build new chunk from  $\bar{v}^1$ 
23:      |  $S_2 \leftarrow C$ 
24:    end if
25:    PROCESSEVENTSEGMENT( $v^1$ ) // Train on segment  $v^1$ 
26:    UPDATECOUNTS( $S$ ) // Update CM counts for all elements in  $S$ 
27:  end for
28:  UPDATEASSOCIATIONS( ${}^\Omega \delta_t$ ) // Update associations for all WM elements
29: end procedure
```

The complete process of updating the WM and performing learning/inference on the CM is given in Algorithm 4.6. This algorithm takes the set of streams as an argument, and is run at each time step, after the PE has processed group G_t . The UPDATESALIENCE function updates the cognitive salience and habituation of each element in S , and the RECOGNITION function returns the recognition level of the given segment. The BUILDCHUNK function wraps a segment in a new chunk, and the CONTIGUOUSCHUNKS function scans the stream for lists of contiguous, same-level chunks. The PROCESSCONTIGCHUNKS function performs learning/inference of higher-level segments (i.e., segments of chunks) and the PROCESSEVENTSEGMENT function performs learning/inference on L_1 event segments.

The UPDATECOUNTS function scans the stream and runs one inference pass on all contiguous events and/or chunks that have not yet had their counts (i.e., weights) updated, and updates the counts of all inferred edges. When all streams have been processed, the UPDATEASSOCIATIONS function creates and/or strengthens associations between all nodes visited in the current time step. The various steps in this process will be explained in more detail throughout the following section.

Algorithm 4.7 Learning and inference of L_1 segments

```

1: procedure PROCESSEVENTSEGMENT( $v^1$ )           // CM learning of segment  $v^1$ 
2:    $i \leftarrow 0$ 
3:    $\delta \leftarrow L_1 \eta_0^0$                    // Set state to  $L_1$  root node
4:   for  $k \leftarrow 2, |v^1|$  do
5:      $\varepsilon \leftarrow v_k^1$ 
6:      $\delta' \leftarrow \text{SEARCHTRANSITION}(\tau(\delta, \varepsilon))$  // Search CM graph for transition
7:     if  $\delta' = \perp$  then
8:        $\delta \leftarrow \text{EXTENDLEVEL}(\delta, \varepsilon)$  // Add tier 0 node to CM graph
9:     else
10:      if  $i < 2$  then
11:         $i \leftarrow i + 1$ 
12:         $\delta \leftarrow \text{ADDNODE}(\delta', i, \varepsilon)$  // Add node at tier  $i$ 
13:      else
14:         $\delta \leftarrow \delta'$ 
15:         $\text{UPDATEPREDICTABILITY}(v^{\text{pred}})$  // Update predictability of segment
16:         $v^1 \theta_k^i \leftarrow \delta$  // Set its terminal node
17:         $L_1 \delta_t \leftarrow \delta$  // Update  $L_1$  state
18:      end if
19:    end if
20:     $k \leftarrow k + 1$ 
21:  end for
22: end procedure

```

When training begins, the model has one level L_1 , with a single *root node* $L_1 \eta_0^0$. When a WM segment v^{L_1} is received, it is passed to L_1 , and Algorithm 4.7 is used for learning and inference. In this algorithm, the SEARCHTRANSITION function looks for the given transition, at the highest tier possible, and returns either the inferred node or \perp . The EXTENDLEVEL function adds a new node at depth k and tier 0 with its value derived from ε , creates an edge for the transition, and returns the new node. The ADDNODE function adds a new node at depth k and tier i with its value derived from ε , and returns the new node. If the tier of the new node matches the tier of δ , an edge is added between δ and the new node. UPDATEPREDICTABILITY calculates a predictability value for the segment v^{pred} ,

as a function of the learned/inferred node’s tier (tier 0 = 0.3, tier 1 = 0.8, tier 2 = 1). The predictability assigned to a given segment is the mean predictability level of all nodes visited during learning/inference of the segment (i.e., all nodes along the inferred CM path). Note that the first event is skipped (i.e., $k \leftarrow 2$), since the objective at L_1 is to encode only the internal shape of the motif, independent of the boundary.

Rather than having the CM maintain its own state after learning/inference, the state is stored with the inferred stream. By storing the state with the stream in WM, rather than in LTM, the CM remains stateless, and is thus able to iteratively process multiple, concurrent streams. Since L_1 segments are passed to the CM as they are being constructed, each call to `PROCESSEVENTSEGMENT` incrementally updates the L_1 state, providing a form of event-level inference. It is important to note that, due to the iterative and incremental presentation of segments to the CM, we do not update edge counts as nodes are added. Rather, the transition counts of all *closed* (i.e., completed) segments are updated in WM, in a single pass, as part of the `UPDATEWM` function (Algorithm 4.6 above).

Learning above L_1 is handled by Algorithm 4.8, which follows essentially the same pattern (as Algorithm 4.7), except that processing deals with sequences of chunks. Because it must now encode the L_1 segment boundaries (which were omitted during event-level learning), this algorithm does not skip the first element in the sequence. `PROCESSCONTIGCHUNKS` takes either an array of contiguous chunks, or a segment containing chunks (as returned by the `CHUNKPARALLELISM` function) as input. The `GETLEVEL` function returns the hierarchical level of the first chunk in A , which it uses as the level for learning/inference. If the input segment suggests a level of form beyond the current capacity of the CM, the `ADDLEVEL` function adds a new level to the top of the CM. The assignment $\varepsilon \leftarrow \beta A_k$ indicates that the element being learned is the boundary event (i.e., first event) of the k^{th} chunk in A . Finally, when the learned/inferred state δ' has been set, across-level links $\iota(a, b)$ are created. The first link, at line 37, is a “terminal link” connecting the inferred boundary δ' to the terminal node of the current chunk A_k . The second, on line 38, is a “cueing link” that connects the terminal node of the *previous chunk* A_{k-1} to the inferred boundary (as demonstrated in Figure 4.15).

Algorithm 4.8 Learning and inference of higher-level segments

```
1: procedure PROCESSCONTIGCHUNKS( $A$ )      // CM learning from chunk sequence  $A$ 
2:    $i \leftarrow 0$ 
3:    $n \leftarrow \text{GETLEVEL}(A_1)$            // Get level from first chunk in  $A$ 
4:   if  $n + 1 > |M|$  then                 // If adding a level will exceed the model's size
5:      $\text{ADDLEVEL}(M_{n+1})$                  // Add a new level to model  $M$ 
6:   end if
7:    $\delta \leftarrow L_{n+1}\eta_0^0$          // Set state to  $L_{n+1}$  root node
8:   for  $k \leftarrow 1, |A|$  do
9:      $\varepsilon \leftarrow \beta A_k$        // Get boundary of chunk  $A_k$ 
10:     $j \leftarrow |A_k|$ 
11:     $\delta' \leftarrow \text{SEARCHTRANSITION}(\tau(\delta, \varepsilon))$  // Search CM graph for boundary transition
12:    if  $\delta' = \perp$  then
13:       $x \leftarrow \text{true}$ 
14:      if  $k = 1$  then
15:         $\Delta \leftarrow \text{GETSIBLINGS}(\delta)$  // Get nodes at same CM depth as  $\delta$ 
16:        for  $\delta^s \in \Delta$  do
17:           $\delta \leftarrow \text{SEARCHTRANSITION}(\tau(\delta^s, \varepsilon))$ 
18:          if  $\delta \neq \perp$  then
19:             $\text{ADDPARENT}(\delta^s, \delta)$  // Transition to sibling found; add parent
20:             $x \leftarrow \text{false}$ 
21:          end if
22:        end for
23:      end if
24:      if  $x = \text{true}$  then
25:         $\delta \leftarrow \text{EXTENDLEVEL}(\delta, \varepsilon)$ 
26:      end if
27:    else
28:      if  $i < 2$  then
29:         $i \leftarrow i + 1$ 
30:         $\delta \leftarrow \text{ADDNODE}(\delta', i, \varepsilon)$ 
31:      else
32:         $\delta \leftarrow \delta'$ 
33:         $\text{UPDATEPREDICTABILITY}(\varepsilon)$ 
34:        if  $\text{ISSEGMENT}(A)$  then // If  $A$  is a segment...
35:           $A\theta_k^i \leftarrow \delta'$  // ...set  $\delta'$  as its terminal node
36:        end if
37:         $\iota(\delta', A_k\theta_j^i)$  // Link inferred state/node to terminal of chunk  $A_k$ 
38:         $\iota(A_{k-1}\theta_j^i, \delta')$  // Link terminal of chunk  $A_{k-1}$  to inferred state/node
39:      end if
40:    end if
41:     $k \leftarrow k + 1$ 
42:  end for
43: end procedure
```

When learning at higher levels an important change is made to the learning algorithm at depth $k = 1$. If the initial search from node δ (line 11) fails to find the transition, the algorithm will repeat the search across all siblings of δ (i.e., nodes of equal depth and tier) at line 18. This routine allows for increased compression, while at the same time preserving important contextual information—i.e., the boundary contour and interval—encoded by the depth $k = 1$ nodes. This can be seen in Figure 4.15, where the two statements of the phrase have the same basic contour, but different boundaries, as a result of their different musical contexts. The first statement, which opens the work, has no context (Schema 0)¹², while the second statement is a repetition of the first, transposed down one scale-step (Schema -). However, since the two contours are otherwise identical, their learned paths merge at depth 2, allowing both phrases to reach the same terminal node. This shared terminal node reflects the musical similarity of the two phrases, allowing MusiCog to treat them as similar musical concepts.

4.2.4 The Production Module (PM)

The knowledge represented by MusiCog's various processing modules, and captured in its memory structures, is essentially *procedural* and *implicit*. It does not represent knowledge of music theory, nor any sort of explicit knowledge about music composition, but is rather an expression of the music psychological principles that inform MusiCog's design. By way of analogy, MusiCog could be compared to a musician with some degree of instrumental training (i.e., it can play sequences of notes), but no explicit knowledge of music theory or composition. Thus, our investigation of the compositional capacities of MusiCog (see Section 5.4) attempts to focus on the architectural factors that influence its musical output. The PM's generative algorithms, in its current state of development, are stochastic¹³. The only notable exception is the top-down/bottom-up pattern of LTM exploration used during generation, which suggests a process of compositional "planning" and execution, discussed in more detail in Section 4.2.4. While this notion of beginning with a higher-level abstract plan could suggest a philosophy of composition, it is also possible to generate a plan based on inference of musical input, as is the case when generating musical "continuations" from MusiCog¹⁴. Thus, although our choice to start new generations using a top-down planning

¹²Strictly speaking, a Schema of zero indicates a repetition. However, we also use the zero Schema in the case of phrases where no contextual information is available, to suggest that no *change* has been detected.

¹³We do not maintain that a stochastic process can explain compositional thinking. This is merely a starting point for investigating MusiCog's compositional tendencies at an architectural level.

¹⁴This is essentially the routine followed by MusiCog when carrying out extended autonomous generations.

process is a compositional decision of sorts, it should be clear that this is not the only way to generate output from MusiCog. Our justification for starting with a fundamentally stochastic method is twofold. First, we wanted to gain some understanding of the musical limitations imposed by the architecture itself, and by its music psychological underpinnings. Second, by establishing a primarily architectural and stochastic mode of generation as a sort of ground truth, future research can focus on the influence of specific music theoretical and compositional knowledge representations and decision processes on the quality of the generated music.

On an architectural level, the PM generates musical output following the principle that music comprehension is supported by the retention of materials in working memory, and that this provides a motivation for composition. Through motivic exploitation—i.e., the use of musical parallelism—composers are able to guide listeners through the musical discourse, introducing thematic materials in an intelligible manner. Models that focus only on reproducing the note transition probabilities of a training corpus generally fail to produce convincing results because they ignore musical parallelism (or achieve it only accidentally); the *IDyOM* model of Wiggins et al. is a case in point [242]. This notion that working memory provides the locus for both compositional intentionality and listener comprehension can be related to Schmidhuber’s theories of compression and creativity [208], and also to ideas from the field of literary composition theory [85, 99], where working memory capacity is thought to be a determining factor on the quality and complexity of written material [166, 167] (as discussed in Section 2.2).

PM generation combines high-level planning from the CM with the integration of local contextual information from WM. There are two basic approaches that can be taken: 1) Generate a high-level plan and fill-in the surface details according to the plan (i.e., “top-down”), and 2) Incrementally infer phrase-level plans, based on WM content, and gradually construct a high-level plan (i.e., “bottom-up”). It has been suggested—for example, in Collins’ synthesis process model [41] (see Section 2.3)—that compositional thinking enlists both processes, used in alternation. During generation, the PM attempts to integrate knowledge from the WM and CM, with the goal of producing well-formed musical segments that reflect an appropriate degree of musical parallelism. Since parallelism is generally evidenced in phrase-level form (see Section 4.2.2), the PM must first determine a phrase structure or “plan” to guide the generation.

Formal Planning

In a “top-down” approach to form generation, the PM first probabilistically creates a high-level CM path $P_d^{L_n}$, of length d , referred to as a *plan*. This is done using the FORWARDPLAN algorithm (Algorithm 4.9), which creates a sequence of boundary nodes starting from an arbitrary input node (generally the L_n root node).

Algorithm 4.9 Predictive generation from a given state

```

1: procedure FORWARDPLAN( $L_n \eta_k^i$ )           // Generate predictive plan given node  $L_n \eta_k^i$ 
2:    $P \leftarrow \emptyset$                        // Create a new, empty path
3:    $d \leftarrow 1$ 
4:    $\Upsilon \leftarrow N^+(L_n \eta_k^i)$          //  $\Upsilon$  gets the passed node's out-neighbours
5:   if  $|\Upsilon| = 0 \wedge i > 0$  then
6:     FORWARDPLAN( $L_n \eta_k^{i-1}$ )           // If no out-neighbours, recurse at subtier
7:   end if
8:   while  $|\Upsilon| > 0$  do
9:      $\delta \leftarrow \text{CHOOSENODE}(\Upsilon)$ 
10:    if GETLEVEL( $\delta$ ) =  $n$  then
11:       $P_d \leftarrow \delta$ 
12:    else
13:      break
14:    end if
15:     $\Upsilon \leftarrow N^+(\delta)$ 
16:    if  $|\Upsilon| = 0 \wedge i > 0$  then
17:       $\Upsilon \leftarrow N^+(\delta^{i-1})$        // If no out-neighbours, try at subtier
18:    end if
19:     $k \leftarrow k + 1$ 
20:     $d \leftarrow d + 1$ 
21:  end while
22: end procedure

```

The algorithm traverses the directed subgraph at level L_n , making weighted probabilistic selections (using the CHOOSENODE function) from the set of out-edges at each step. The chosen edge's incident node is added to the end of plan $P_d^{L_n}$, and the process is iterated until a terminal node is reached. The condition at line 10 in the algorithm is used to terminate the traversal, in the event that an L_{n+1} cueing link is chosen, rather than an L_n child edge.

This terminating condition is illustrated in Figure 4.17, where we see that cueing links can identify terminal nodes (i.e., segment endings) that are not necessarily leaf nodes. Here, the two out-edges of node **C** connect to a child node and a subsequent boundary

node, with weights 1 and 5 respectively, indicating a high probability that the node will terminate the segment (i.e., by following the cueing link to the L_2 boundary node **E** rather than continuing along L_1 to node **G**).

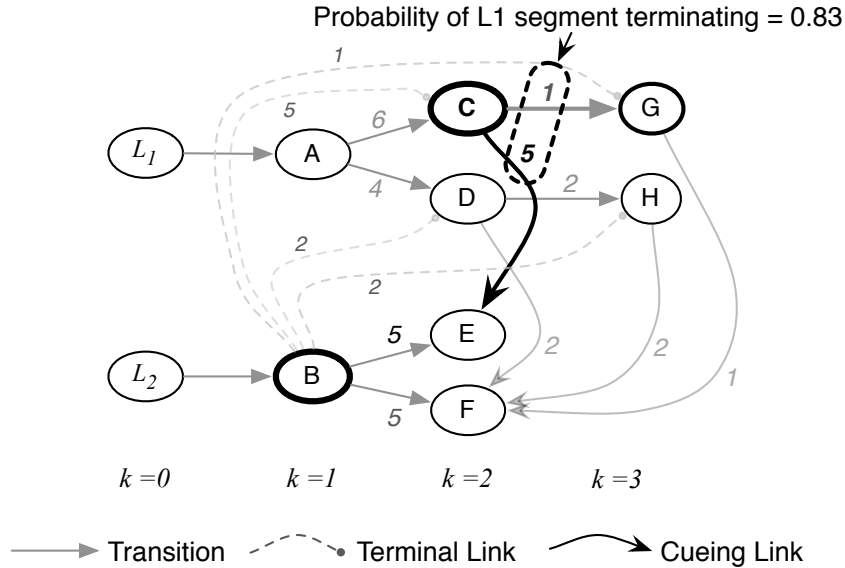


Figure 4.17: A hypothetical CM graph in which the out-edge weights indicate a high-probability transition to an L_2 boundary node, thus terminating the L_1 segment. This property of the CM can be used to identify terminal nodes that are not necessarily leaf nodes.

Once a high-level plan $P_d^{L_n}$ has been created, the PM then proceeds to “unwind” a series of level L_{n-1} paths, one for each node in $P_d^{L_n}$, using the PATHFORBOUNDARY algorithm (Algorithm 4.10). This algorithm must account for a number of factors, as illustrated in Figure 4.18. Here we see that there are three terminals—**I**, **J**, and **G**—connected to node **C**, from which an L_{n-1} path is to be derived. We also see that the L_n plan continues to node **H**. The task is therefore to find a path through L_{n-1} that connects node **C** to node **H**. In the algorithm, the GETTERMINALS function returns the set of terminal nodes T connected to the given boundary $L_n \eta_k^i$ (i.e., node **C**). The PLANFROMTERMINAL algorithm (Algorithm 4.11) then selects a path leading to each terminal T_j (i.e., from the root node). The path to T_j is stored at S_j , and its probability is stored at \mathbf{p}_j . Vector \mathbf{t} stores the probabilities of selecting the terminals themselves from L_n , and vector \mathbf{c} is a binary vector used to filter out terminals that do not connect to the given “goal” node $L_n \eta_{k+1}^i$ (i.e., node **H**).

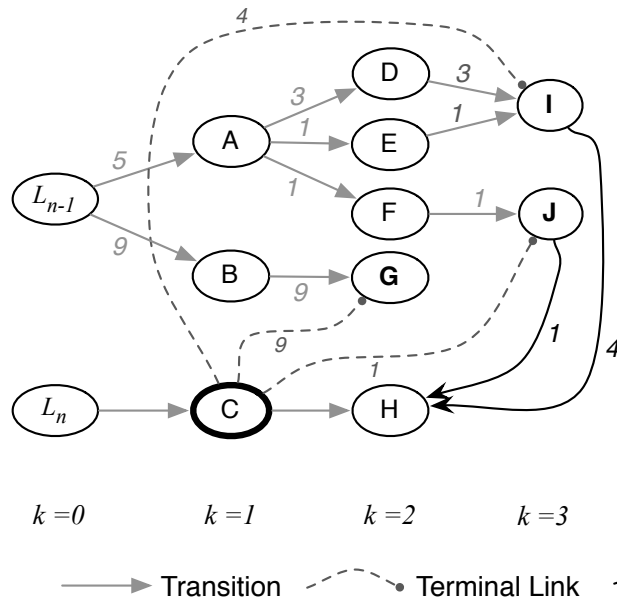
The final probability distribution is the cross-product of \mathbf{p} , \mathbf{t} , and \mathbf{c} , from which the WEIGHTEDPROBABILISTICINDEX function makes a stochastic selection and returns the

Algorithm 4.10 Selection of L_{n-1} path given L_n boundary

```
1: function PATHFORBOUNDARY( $L_n\eta_k^i, L_n\eta_{k+1}^i$ ) // Get sublevel path given boundary
//  $L_n\eta_k^i$  and "goal"  $L_n\eta_{k+1}^i$ 
2:    $T \leftarrow$  GETTERMINALS( $L_n\eta_k^i$ )
3:    $S \leftarrow \emptyset$ 
4:   for  $j \leftarrow i, |T|$  do
5:      $S_j \leftarrow$  PLANFROMTERMINAL( $T_j$ )
6:      $\mathbf{p}_j \leftarrow P(S_j)$  // Get probability of sublevel path  $S_j$ 
7:      $\mathbf{t}_j \leftarrow P(T_j)$  // Get probability of terminal  $T_j$ 
8:     if  $L_n\eta_{k+1}^i \in N^+(T)$  then // If  $L_n\eta_{k+1}^i$  is a target of terminal  $T$ 
9:       |  $\mathbf{c}_j \leftarrow 1$ 
10:    else
11:      |  $\mathbf{c}_j \leftarrow 0$ 
12:    end if
13:  end for
14:   $\mathbf{x} \leftarrow \mathbf{p} \times \mathbf{t} \times \mathbf{c}$ 
15:   $y \leftarrow$  WEIGHTEDPROBABILISTICINDEX( $\mathbf{x}$ )
16:  return  $S_y$ 
17: end function
```

chosen index y . This index is used to retrieve the corresponding path from S . In the diagram we see that the highest-probability path $S_3 = 64\%$ has a 0% probability of selection, due to the fact that it does not connect to the goal node \mathbf{H} , and is filtered out by vector \mathbf{c} .

The PLANFROMTERMINAL algorithm, which is used to build the set of possible paths S , takes a terminal node $L_{n-1}\theta_k^i$ as an argument, and iteratively backtracks from depth k to depth 1, stochastically selecting from the parent nodes at each step, and inserting the chosen node at the start of path A . The stochastic selection is done using the CHOOSE-NODE function (as in the FORWARDPLAN algorithm), except that in this case the selection is made from the passed node's set of in-neighbours (i.e., parent nodes), and termination is handled by reaching the root node.



Chosen path for each terminal:

$$S_1 = (A, D, I)$$

$$S_2 = (A, F, J)$$

$$S_3 = (B, G)$$

Probabilities of paths, terminals, and targets:

$$\mathbf{p}(S_1, S_2, S_3) = (0.16, 0.07, 0.64)$$

$$\mathbf{t}(I, J, G) = (0.29, 0.07, 0.64)$$

$$\mathbf{c}(I, J, G) = (1, 1, 0)$$

$$\mathbf{p} \times \mathbf{t} \times \mathbf{c} = (0.046, 0.005, 0)$$

Terminal selection probabilities:

$$P(I, J, G) = (0.9, 0.1, 0)$$

$k=0$ $k=1$ $k=2$ $k=3$

→ Transition • Terminal Link ↪ Cueing Link

Figure 4.18: Selection of an L_{n-1} segment from L_n boundary **C**. The algorithm must account for L_{n-1} segment probabilities, terminal probabilities, and connectivity to **H** via cueing links.

Algorithm 4.11 Generate path to a given terminal

```

1: function PLANFROMTERMINAL( $L_n \theta_k^i$ ) // Get path to terminal  $L_n \theta_k^i$ 
2:    $A \leftarrow \emptyset$ 
3:    $\rho \leftarrow N^-(L_n \theta_k^i)$  // Get the passed node's parents
4:   if  $|\rho| = 0 \wedge i > 0$  then
5:     | PLANFROMTERMINAL( $L_n \theta_k^{i-1}$ ) // If no parents, try at subtier
6:   end if
7:    $A_1 \leftarrow L_n \theta_k^i$ 
8:   while  $|\rho| > 0$  do
9:     |  $\delta^i \leftarrow \text{CHOOSENODE}(\rho)$ 
10:    | for  $j \leftarrow 1, k-1$  do  $A_{j+1} \leftarrow A_j$  // Copy nodes in A to next index
11:    | end for
12:    |  $A_1 \leftarrow \delta^i$  // Insert  $\delta^i$  at start of A
13:    | if  $k < 2$  then
14:    | | break
15:    | else
16:    | |  $\rho \leftarrow N^-(\delta^i)$  // Get parent's of  $\delta^i$ 
17:    | | end if
18:    | | if  $|\rho| = 0 \wedge i > 0$  then
19:    | | |  $\rho \leftarrow N^-(\delta^{i-1})$  // If no parents, try at subtier
20:    | | end if
21:    | end while
22:   return A
23: end function

```

Once the lower-level plan has been created in this manner, the same process can be carried out recursively from $P_1^{L_{n-1}}$ —i.e., selecting a L_{n-2} terminal of $P_1^{L_{n-1}}$, then building a plan by backtracking from $L_{n-2}\theta_k^i$ —until a complete hierarchy is defined. This procedure can be followed for all nodes $\{1 \dots d\}$ in $P_d^{L_n}$, resulting in the creation of a series of L_{n-1} plans, making it possible to carry out top-down generation of large-scale hierarchical forms. This approach can be used with a CM of arbitrary size.

In a ‘bottom-up’ approach, form generation begins from the stream state $^S\delta_t$ (which reflects the contents of WM), then stochastically unfolds the higher-level form by selecting transitions on each level, as required, using the CHOOSENODE function. In this case, CHOOSENODE is given the union of out-edges and cueing links from the current state node, allowing segment termination to be handled stochastically, as it is in the FORWARDPLAN algorithm (Algorithm 4.9). Thus, when L_1 generation reaches a terminal node, a boundary for generating the next segment is created by selecting a transition at L_2 . Likewise, when L_2 reaches a terminal node, a new phrase boundary is created by selecting a transition at L_3 , and so on. When bottom-up continuations of this sort occur above L_1 , the stochastic selection at L_n incorporates the L_{n-1} evidence provided by *cueing links* (described in Section 4.2.3) from the current L_n node’s terminal nodes, in addition to the information provided by its child edges. This process is illustrated in Figure 4.19, where the choice between transitions (B,E) and (B,F) is equiprobable on L_2 , but can be shown to favour transition (B,E) when the L_1 state information (node C) is included. Here it can be seen that cueing link $\iota(\mathbf{C}, \mathbf{E})$ is the only link to node E¹⁵.

In order to integrate as much “planning” as possible into the bottom-up approach, the PM will always generate a new L_2 plan when the current plan is finished. After the last node of the current L_2 plan has been used for generation, a new L_2 plan is generated, using the FORWARDPLAN algorithm. This ensures that there is always a current plan ready to guide L_1 generation.

Parallelism, Motivic Exploitation, and Segment Output

As was mentioned in the PM introduction (on p. 111), PM generation is designed to acknowledge the role of working memory in the comprehension of music. In order to do this, the generation algorithms attempt to exploit previously used motivic material. Through the learning process, parallelisms in the training material are encoded implicitly in the CM data

¹⁵Although there is some probability that node C could cue node F, via node G, this future probability is not currently considered in MusiCog.

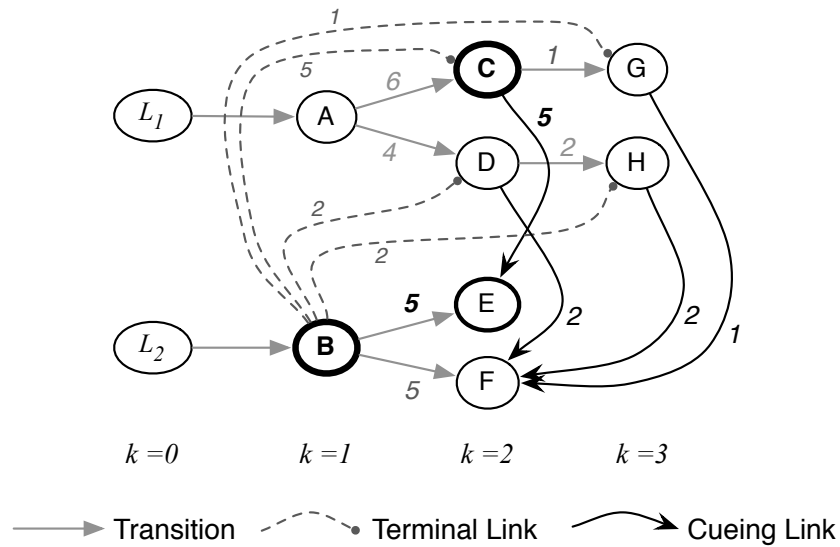


Figure 4.19: A detailed Invariance tier view of a hypothetical CM for pitch information showing weights for transitions, terminal links, and cueing links. Bold nodes indicate the current state. Although transitions (B,E) and (B,F) are equiprobable, (B,E) has greater support from the L1 state, via its cueing link.

structure, via the ratio of boundaries to unique terminals. This can be seen in Figure 4.20a, where the sharing of terminal **X** by boundaries **T** and **U** indicates the parallelism. However, as new material is learned, compression in the CM causes these clear points of parallelism to be obscured, as in Figure 4.20b. Here, the parallelism in the melody is expressed through the sharing of terminal **Y** between boundaries **S** and **T**. However, due to previous training on melody 4.20a, stochastic selection of terminal **Y** for boundary **T** can only occur at chance level¹⁶.

Of course, the connectivity of the CM still represents the parallelisms expressed in both example melodies, even if their probability of being selected has been reduced. What is needed, therefore, is a method for supporting the selection of such parallelisms, based on the developing musical context. One way to do this is to examine the contents of WM as the generation progresses and to search for intersections between the terminals currently held in WM and the terminals linked to the current CM boundary state. Looking again at

¹⁶Of course, in this specific example, Invariance tier information can help distinguish the two melodies, and retain the capacity to produce these parallelisms during generation. However, it is likely that further training will eventually eliminate such clear points of differentiation.

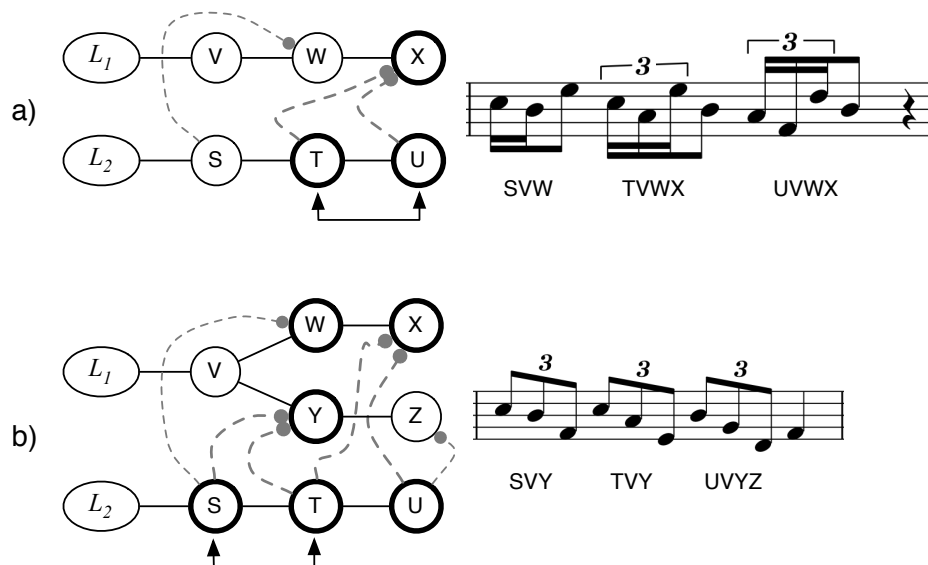


Figure 4.20: The melodic fragment represented in CM graph **a** expresses a clear parallelism via the sharing of terminal **X** with boundaries **T** and **U**. However, the probability of generating this parallelism is jeopardized by subsequent learning of melody **b**.

Figure 4.20, let us imagine that the generation process has recently produced the segment **(S,V,Y)**, so that the WM currently contains a reference to terminal **Y**. The boundary for the next generation is **T**, which has terminal links to **Y** and **X**. In order to exploit the parallelism represented in the model, with support from WM, we could simply choose terminal **Y** for the new generation. Of course, an obvious problem with this approach is that it ignores the model's capacity for change by forcing the parallelism (further, it will usually be the case that more than one terminal is supported in WM). For this reason, rather than automatically using the terminal stored in WM (i.e., node **Y**), we instead bias the choice toward this terminal by adding the WM copy to the set of possible choices—i.e., so that the node representing the parallelism is represented *twice* in the set of choices, increasing its probability of being selected. Thus, the set of possible terminals for selection by node **T** will be **{Y,Y,X}**, increasing the probability of selecting the parallelism represented by terminal **Y**, while retaining the possibility of selecting the non-parallel terminal **X**.

When using terminals from WM, the chosen terminal is paired with a boundary drawn from the current L_2 plan, and an L_1 path for the terminal is extracted from the CM using the PLANFROMTERMINAL algorithm (Algorithm 4.11). In this way, the generated segment draws on both WM and LTM content, and involves both top-down and bottom-up factors. Since the topology of the CM guarantees that all possible paths to a given terminal share

the same contour, this process allows generation to maintain parallelism, while at the same time promoting variety. Due to this mixing of WM and LTM elements, it is possible that the pairing of an L_2 boundary with an L_1 path may not have appeared during training, allowing MusiCog to introduce novel segments not present in the training corpus, and thus enabling a form of creativity.

Once an L_2 plan is in place, the PM can proceed to generate an output segment. Although outputs are presented on a note-to-note basis, the PM always generates in segments (or *motives*), not individual events. All events in the most recently generated segment are kept in a FIFO buffer and extracted sequentially as the segment is rendered (i.e., during playback). The PM also keeps the current L_2 plan on a stack, so that the top of the stack always contains the *next* boundary for generation. When a new segment is required, an L_2 boundary is popped off the stack, and a new L_1 path is generated using the PATHFORBOUNDARY algorithm (Algorithm 4.10)—or using the L_2 boundary, a terminal from WM, and the PLANFROMTERMINAL algorithm, as discussed above. In order to allow generated material to adapt to the current musical context, only Invariance tier nodes are used for generating output events. If the inference process returns an Identity node as the current WM state (and thus the context for continuation), the PM extracts the node's associated Invariance node (see Figure 4.11) before generating a continuation. The PM uses the most recent event in the generated stream S_1 to determine the context for the segment to be generated, and extracts Invariance information from the nodes in the generated path, which it converts to a series of events, wrapped in a segment. In the event that the stream is empty, the PM must first establish the musical context. To do this, it will start generation by selecting an L_2 , depth 1, Identity tier node (i.e., to establish an explicit, context-free boundary), after which Invariance tier nodes will be used.

Feedback: Acting and Perceiving

In order to complete the agent metaphor, MusiCog must not only *act* in its world, it must also perceive the results of its actions. To realize this, segments generated by the PM are fed back into the PE. This is an important step, because it allows the model to evaluate its actions, and modify its behaviour accordingly, providing a rudimentary form of *intentionality*. In the current state of MusiCog's development, this intentionality is limited to a simple goal: to generate phrases with well-defined segment boundaries, as indicated by the PE's grouping mechanisms.

There are two ways that this can be achieved. One option is to evaluate potential generations before output, using the PE's segmentation algorithm, and to reject segments that either continue the preceding segment without triggering a segmentation response, or that cause a segmentation *within* the generated segment. The problem with this approach is that, in certain CM contexts, there may be a large number of potential generations to evaluate, with no guarantee that an appropriate segment will be found. Thus it is possible that, even after a lengthy search, the criteria may not be met, leading to the generation of ill-formed material. The other option is to adapt the PM's planned generation according to the segmentation responses triggered in the PE. In this approach, the PM's pending boundary is offset to account for the PE's alteration of the melodic segment structure. This process is illustrated in Figure 4.21.

Here we see that the PM's original plan (i.e., the L_2 path) was to generate two segments, the second of which was to begin two semitones above the first. However, the PE detected a boundary half-way through the PM's first generated segment, altering the musical context for the pending generation. If the PM were to proceed without modifying its plan, the "+2" interval would be applied to the new (and unexpected) boundary "E," causing the system to generate the sequence: (C, D) (E, F) (F#, ...). While this result introduces a degree of novelty, which may be of some value, it does not follow the structure of the original plan. By modifying the plan in response to the newly perceived boundary, the PM is able to generate a sequence that maintains the intended structure: (C, D) (E, F) (D, ...). It should be noted that this process of adapting the planned structure to accommodate for changes in the perceived structure is perhaps even more crucial with regard to rhythmic generation, where changes in boundary relationships can lead to overlapping segments, or segments

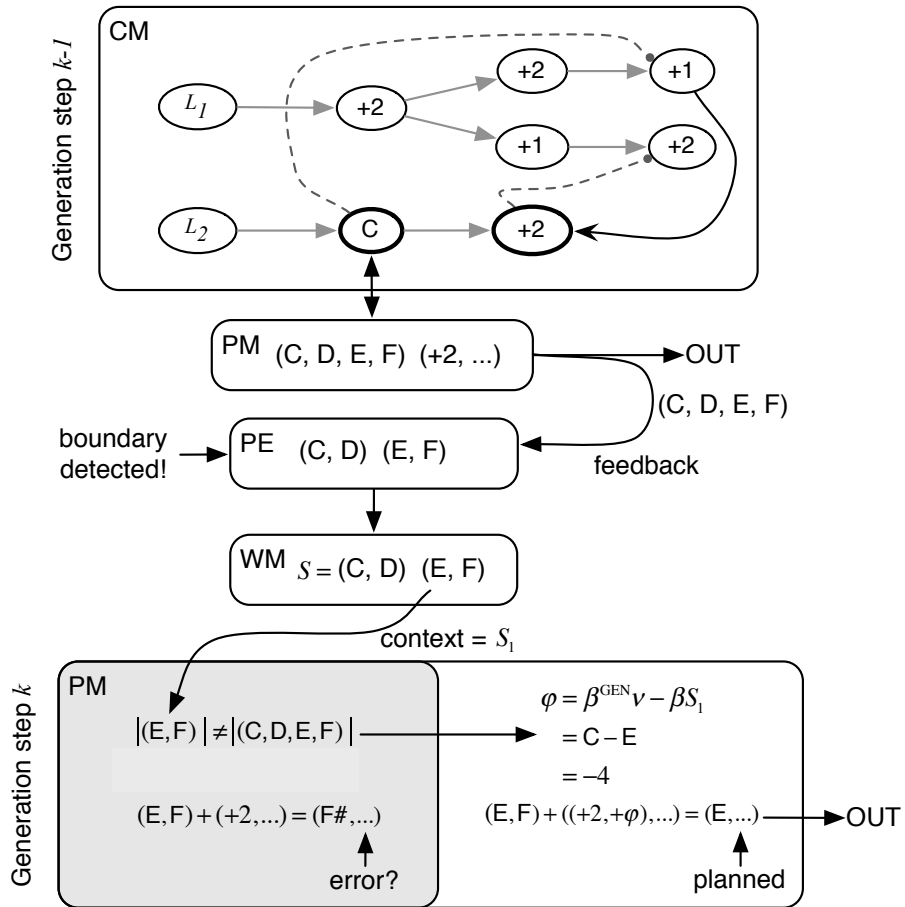


Figure 4.21: During generation the PE detects a segment boundary where the PM had not planned one. The PM responds by adjusting its next output according to the difference φ between the last generated segment's boundary event $\beta^{\text{GEN}} \nu$ and the boundary event of the context segment βS_1 .

separated by unintended pauses. Since the effort to satisfy the PE's segmentation mechanism during PM generation may lead to potentially lengthy and fruitless searches, we have chosen to implement the second approach, so that the PM responds adaptively to the perceived segment structure. It would, of course, also be possible to implement *both* options, so that the PM would search for an appropriate segment in collaboration with the PE, and would only modify the original plan if no solution was found.

As a final step, the mode/tonality induction function described in Section 4.2.1 is used to quantize the pitch content of the generated segment to the inferred mode/tonality. A threshold on the induction function's confidence rating (see p. 89) is used to enable/disable pitch quantization, so that it will only be applied when the PE's confidence in the inferred

mode/tonality is relatively high. In this way, the system can quantize diatonic passages in an appropriate manner, while retaining the ability to generate more chromatic passages by disabling pitch quantization. When applying pitch quantization we consider two factors: 1) the estimated mode/tonality of the training corpus, and 2) the estimated mode/tonality of the newly generated segment. Since our intention is to model only “musically naïve” composition at this stage, we take a perception-based approach when determining the initial mode/tonality. First we estimate the mode/tonality for the newly generated segment; i.e., MusiCog generates an initial melodic segment, performs mode/tonality induction, and decides on the mode/tonality based on the PE’s response. If the PE’s confidence in a specific mode/tonality passes a given threshold, the segment is quantized, and the induced mode/tonality is established for the remaining melodic generation. From this point forward MusiCog only decides whether or not to apply pitch quantization to subsequently generated segments. In deciding whether quantization should be applied, we consider the PE’s confidence in the mode/tonality of the new segment, and the mean confidence induced from training on the original corpus. If the average of these two confidence ratings falls below a threshold value (0.1 in our implementation), the segment will not be quantized. This can be seen as a simple form of musical intentionality; i.e., if the new segment is highly chromatic, this is assumed to be a deliberate compositional goal, and quantization is disabled, leaving the chromaticism intact.

4.2.5 Handling Rhythmic Information

The above description of MusiCog’s learning and generation algorithms discussed only the handling of pitch material. It is important to note that rhythmic processing is carried out in the same manner, and utilizes the same learning, inference, and generation mechanisms as pitch processing. In the current implementation, we represent pitch and rhythm information in two separate CM models. Since the PE uses both pitch and rhythmic information during segmentation, the most recent segment S_1 can be passed to both models for learning and inference at each time step. The same is true for higher-level processing of WM chunks. As transitions are learned/inferred in each model, weighted association links are used to build a statistical representation of the relationship between the pitch and rhythm models. During generation, we process pitch information first, and fit rhythmic representations to the planned pitch structure. For example, if an L_1 path is generated from a depth $k = 3$ terminal node in the pitch model, we will likewise generate from a depth $k = 3$ terminal in the rhythm model, whenever possible (i.e., as long as a rhythm node at the desired

depth is reachable from the current state). The rhythm terminal can be selected probabilistically, via association to the chosen pitch terminal, but this is only done in cases where the planned rhythmic structure does not correlate with the pitch structure (i.e., where a planned rhythmic path contains greater or fewer nodes than the planned pitch path). Because MusiCog is a feedback system, inference of previously generated outputs ensures that the states of the two models remain musically related.

4.3 Implementation Details

MusiCog is implemented as an Objective-C framework for Mac OS X, and also as a MaxMSP external. The MusiCog framework makes use of Objective-C 2.0's "Automatic Reference Counting," and for this reason, is 64-bit only. Because the Max external makes calls to the MusiCog framework, the Max external is also limited to the 64-bit Mac platform, running on Max 6.1 or later. MusiCog is also the learning/generation system for our computer-assisted composition application *ManuScore* [163], discussed in Chapter 7.

The framework and Max external can be downloaded here:

<http://www.sfu.ca/~jbmaxwel/MusiCog/downloads.html>.

Chapter 5

MusiCog in Practice

MusiCog was designed with the practical application of Computer-Assisted Composition (CAC) in mind. For this reason, wherever possible, we will demonstrate the functionality of its component modules through practical musical examples. Although we will demonstrate the functionality of all modules, our quantitative testing will focus on monophonic/melodic generation in the PM.

5.1 The Perception Module

The primary functions of the PE that are directly applicable in a CAC context are stream/voice-separation, low-level melodic segmentation, and mode/tonality induction.

5.1.1 PE Stream/Voice-Separation

Figure 5.1 gives an example of voice-separation in the PE, taken from the opening of Bach's Choral, "Freu dich sehr," from Part 1 of his BWV 70 Cantata. It is worth pointing out that, as a cognitive model, MusiCog's voice-separation algorithm does not include any special system for maintaining consistent "voice numbers"; a concept derived primarily from music notation. Aside from influences of timbre and localization, which support stream segregation through psychoacoustic means (see Section 3.2), the notion of maintaining consistent voices is fundamentally theoretical, not perceptual [32]. Thus, perceptual streams have no independent identity, aside from their content at a given moment.

The image displays a musical score for the Part 1 Chorale from Bach's BWV 70, illustrating PE voice-separation. It is divided into two systems, each with four staves. The first system shows four streams: Stream 1 (treble clef), Stream 2 (treble clef), Stream 3 (bass clef), and Stream 4 (bass clef). Stream 3 and Stream 4 are labeled 'unison' and have a '+1' interval bracketed between them. Stream 4 has two '+5' interval brackets. The second system starts at measure 7 and shows a 'voice-crossing' between Stream 3 and Stream 4, with a '+5' interval bracketed between them.

Figure 5.1: PE voice-separation of the Part 1 Chorale from Bach's BWV 70. Since there is no concept of “voice number,” unison voices are assigned to only one stream, and will not be duplicated across streams, as is the case with the two occurrences of G3 (outlined).

In a related manner, since MusiCog does not model the influence of timbre on voice-separation, it cannot discriminate unison voices. This can be seen in Figure 5.1, where pitches that are shared across parts—in this case stream 3 and stream 4, where the original score indicates a unison on G3—cannot be “doubled” by MusiCog’s voice assignment process. For this reason, one of the voices is replaced with a rest. In the second occurrence of this doubling, at the bottom of the page, a voice-crossing occurs, as a result of stream 4 “stealing” the pitch from stream 3. Looking at the preceding music in both streams, it can be seen that the +5 interval motion in stream 4 has occurred more frequently than the +1 motion in stream 3, suggesting that it is probably the *predictability* value that has caused the voice-crossing by favouring the more probable +5 motion in stream 4 over the +1 in stream 3. Since the voices do not, strictly speaking, *cross* at this point, the voice-crossing cost has no influence on the result. It is worth noting that it would be trivial to force MusiCog

to duplicate the shared voice in such situations, but we did not consider this a priority at this time.

The image displays a musical score for Bach's BWV 846 Fugue, illustrating PE voice-separation. The score is organized into four streams: Stream 1 (treble clef), Stream 2 (treble clef), Stream 3 (treble clef), and Stream 4 (bass clef). The first system shows a 'unison' annotation with arrows pointing to overlapping notes in Stream 1 and Stream 3. The second system, starting at measure 5, shows a 'voice-crossing' annotation with arrows pointing to overlapping notes between Stream 1 and Stream 3. The third system, starting at measure 8, also shows a 'voice-crossing' annotation with arrows pointing to overlapping notes between Stream 1 and Stream 3. Motives are highlighted with grey shading throughout the score.

Figure 5.2: PE voice-separation of complex polyphonic material from Bach's BWV 846 Fugue (motives highlighted manually).

A more complex example of voice-separation is shown in Figure 5.2, this time using the C Major Fugue, BWV 846, from Bach's Well-Tempered Klavier. Of note are the voice-crossings at m. 6, in the second system, and m. 9 in the third system. The first instance, at

m. 6, appears to be related to the greedy assignment algorithm, which removes voices from the selection process as they are assigned. Since stream 4 is sustaining at this onset time, it is not included in the cost calculation. For the remaining voices, the lower-cost transitions (G4, A4) in stream 2 and (C4, D4) in stream 1 are assigned first, leaving only the (A3, F4) transition for stream 3. In the second instance, at m. 9, the crossing is likely due to a similar problem, in which the maximum proximity transition (D, D) is assigned to stream 4, leaving the transition (E, C) as the best remaining choice for stream 1. A more thorough approach that tested all possible pairings would solve these problems. Nevertheless, it is certainly worth noting that the PE performs reasonably well on this complex voice-separation task, without the benefit of iterative, offline processing, or a look-ahead function, as is common in voice-separation algorithms [32, 113]. It is also worth noting that all statements of the primary motive appear intact (highlighted in grey by the authors), without being disrupted by voice-crossing problems or rests (i.e., as a result of unison voices). Of course, credit for this goes primarily to Bach for composing a perceptually clear polyphonic texture, but it is important that MusiCog is able to represent these motivic patterns correctly, so that they may lead to the construction of well-formed representations in LTM.

5.1.2 PE Low-level Boundary Detection

Figures 5.3 to 5.5 show the PE's low-level segmentation of three different melodic passages; Bach's BWV 846 Fugue (melodic surface only¹), Mozart's 40th Symphony, and Maxwell's work for flute solo, *Invidere*. In the Bach and Maxwell examples, the cohesion tolerance φ (Section 4.2.1, Equation 4.5) was set to zero. The Mozart example, however, also shows the effect of increasing the cohesion tolerance.

Although there is a considerable contrast of melodic styles across the three works, examination of the melodic segments produced reveals a syntactic continuity across the three examples, with pitch direction changes and rhythmic augmentations frequently associated with the melodic boundaries. Since the PE is intended to model strictly bottom-up processes, this is the expected behaviour, but it also reveals an important motivation behind our choice to take an integrated approach to building a CAC agent. Through the use of the PE, we can ensure that MusiCog learns from a syntactically consistent vocabulary of musical statements. Consequently, unlike in conventional Markov models, the LTM is not strictly

¹This is a monophonic arrangement made by the authors for the purposes of testing segmentation and chunking.

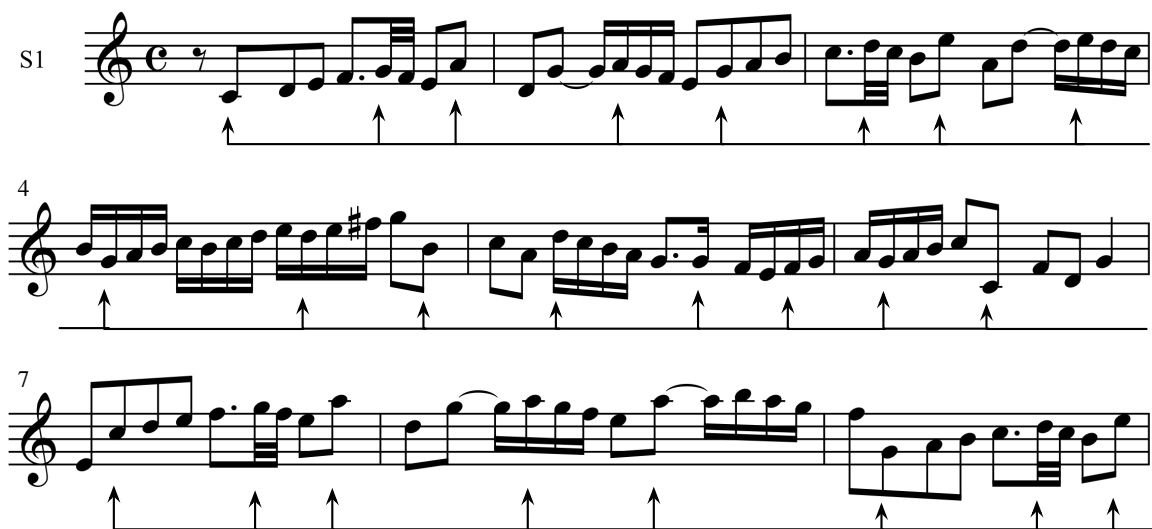


Figure 5.3: Low-level boundary detection for Bach's BWV 846 Fugue (monophonic arrangement).

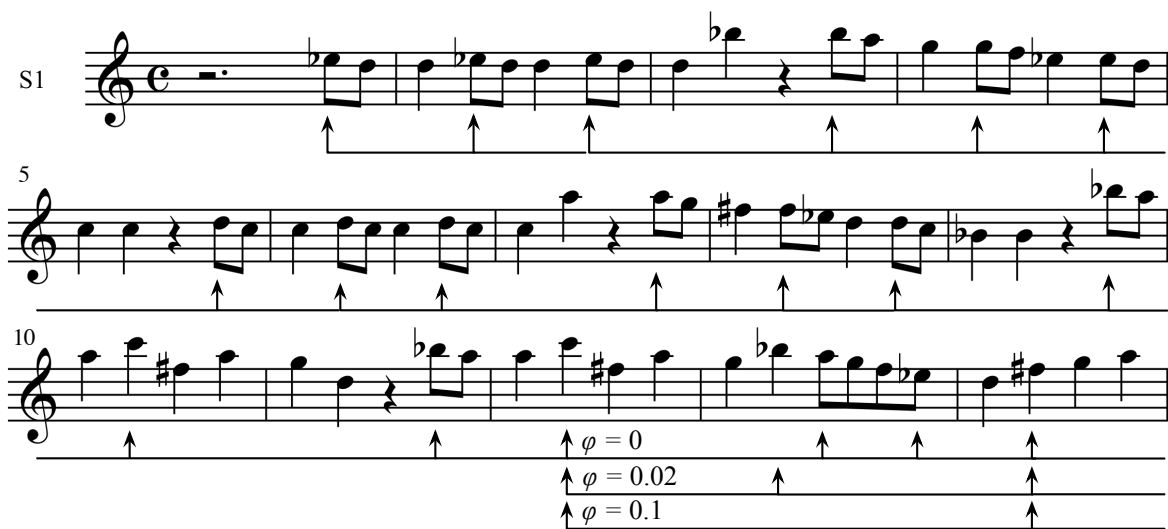


Figure 5.4: Low-level boundary detection for the main theme from Mozart's 40th Symphony. At the end of the last system we see the effect of increasing the cohesion tolerance. It is worth noting that it is not a simple case of removing boundaries as the tolerance increases, but rather that the syntactical content of the segments is also altered.

Figure 5.5: Low-level boundary detection of Maxwell's *Invidere*, for flute solo.

learning the variations of a random variable, but is rather learning the sequential dependencies between tokens in a structured vocabulary. This focus on learning from structured units of input effectively transposes the tokens of musical meaning in MusiCog from the individual pitches themselves to the series of well-formed motives. By analogy to natural language, this is like associating meaning with words rather than individual letters.

In the last system of the Mozart example in Figure 5.4, we see the effect that increasing the cohesion tolerance φ has on the segmentation process. It is important to note that it is not a simple case of removing boundaries. Although boundaries are indeed removed, in some cases the locations of boundaries are also altered, as can be seen in the shift from $\varphi = 0$ to $\varphi = 0.02$. This is in large part due to the tolerance function, which uses a percentage of the standard deviation of cohesion, and is therefore quite sensitive to the specific musical context, but also to the complexities of the rule-based segmentation model itself.

5.1.3 PE Induction of Mode and Tonal Centre

As explained in Section 4.2.1 (p. 87), MusiCog's mode and tonality induction functions do not focus on determining the music theoretical "tonic," *per se*, but rather on finding the best fitting scale formation (i.e., mode) and tonal centre for a given stream or segment. In particular, our interest is in providing the PM with a guide for pitch quantization, so that its interval-based generation process does not wander into inappropriate pitch areas. During processing, induction is performed at the stream and segment levels, but not at the level of individual events. As a result, the induction tends to follow the segment structure quite closely, as can be seen in Figure 5.6. In this example, included to demonstrate the use

of the “confidence” value as a method for detecting chromatic passages, we see that the top passage is correctly identified as C Major, with a brief shift to F Lydian (i.e., the modal equivalent) part-way through, and that this is done with a high level of confidence. In the lower passage, on the other hand, confidence decreases dramatically, and the mode and tonal centre are somewhat ambiguously identified as C Major and C Mixolydian. In this case, the low confidence rating can be used to indicate the presence of chromatic, and tonally ambiguous material.

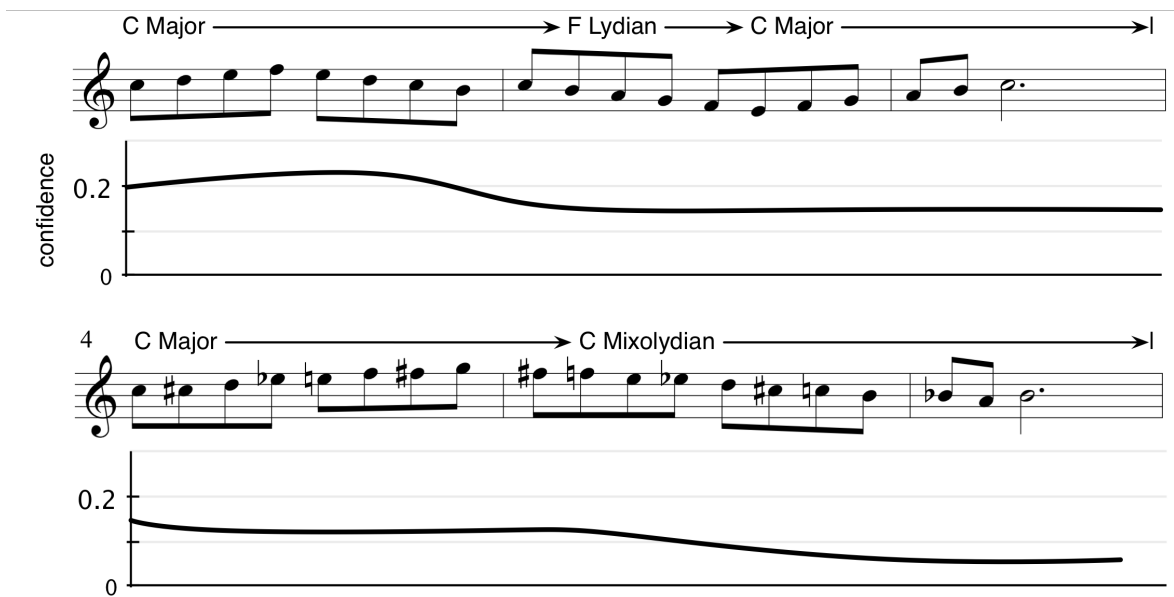


Figure 5.6: A test of the PE's mode and tonal centre induction function. The top passage in C Major invokes a high confidence rating, whereas the lower chromatic passage shows significantly reduced confidence; a behaviour that can be used for the detection of chromatic material.

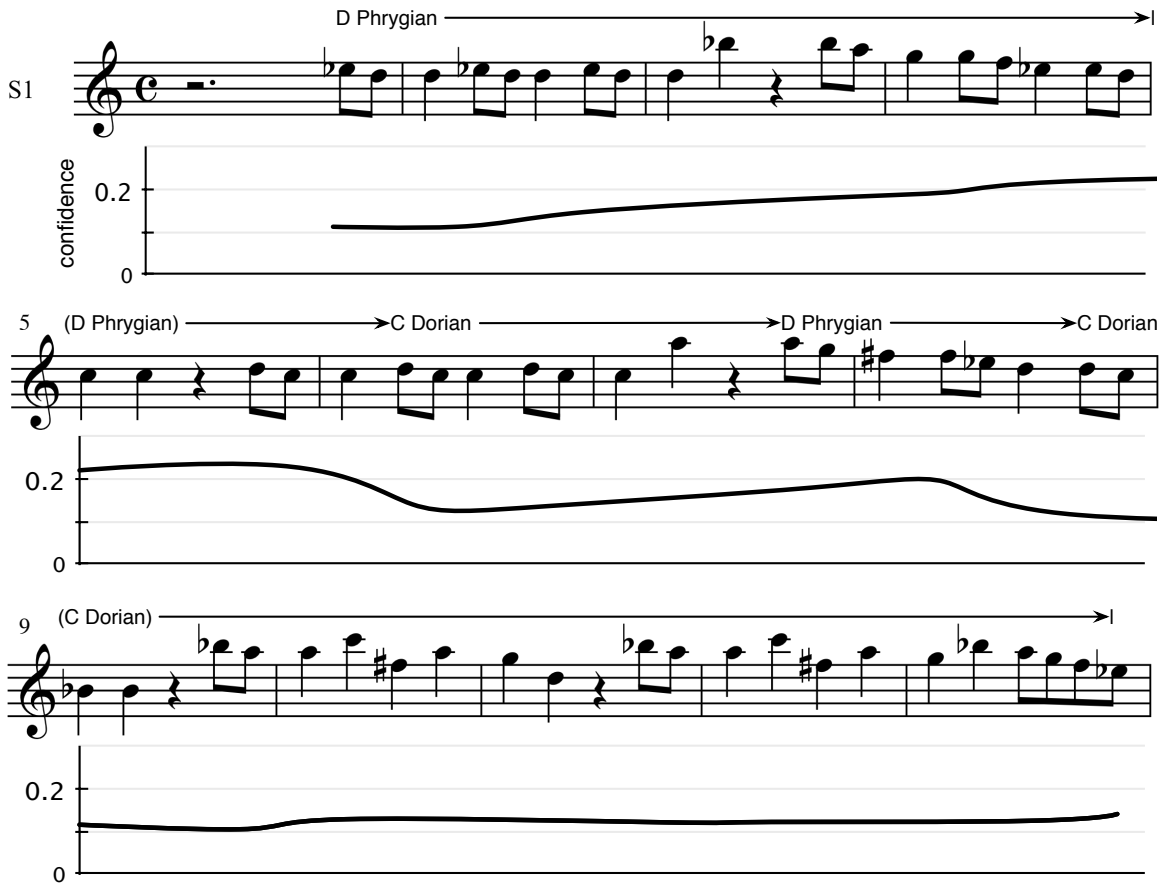


Figure 5.7: Mode and tonal centre induction of the opening melody from Mozart's 40th Symphony. The tonal centre settles on C Dorian, which is the modal equivalent of the correct key, G Minor. Note the decrease in confidence following the introduction of the chromatic F \sharp (and subsequent augmented 3rd interval) at m. 8.

5.2 The Working Memory Module

The WM supports CAC through “listening” and generation, and also through the process of chunking, which allows MusiCog to learn hierarchical/non-adjacent temporal dependencies in LTM. An important factor underlying these functions is the cognitive salience value, and its evolution through time.

5.2.1 WM Cognitive Salience and WM Retention

Figure 5.8 shows the evolution of the cognitive salience value for the opening motive in the Mozart melody (E \flat , D, D). The example shows the final training pass, so that MusiCog has

learned the segment to the Identity tier at this point, as indicated by the high initial salience value. The salience falls to zero in m. 8 as a result of its lack of parallelism with the current segment (F, E \flat , D) and the influence of habituation. The segment is released from WM in the following measure (m. 9). We can also see a period of maintained salience in m. 6, attributable to an increase in parallelism arising from the imitative restatement (D, C, C) of the opening motive.

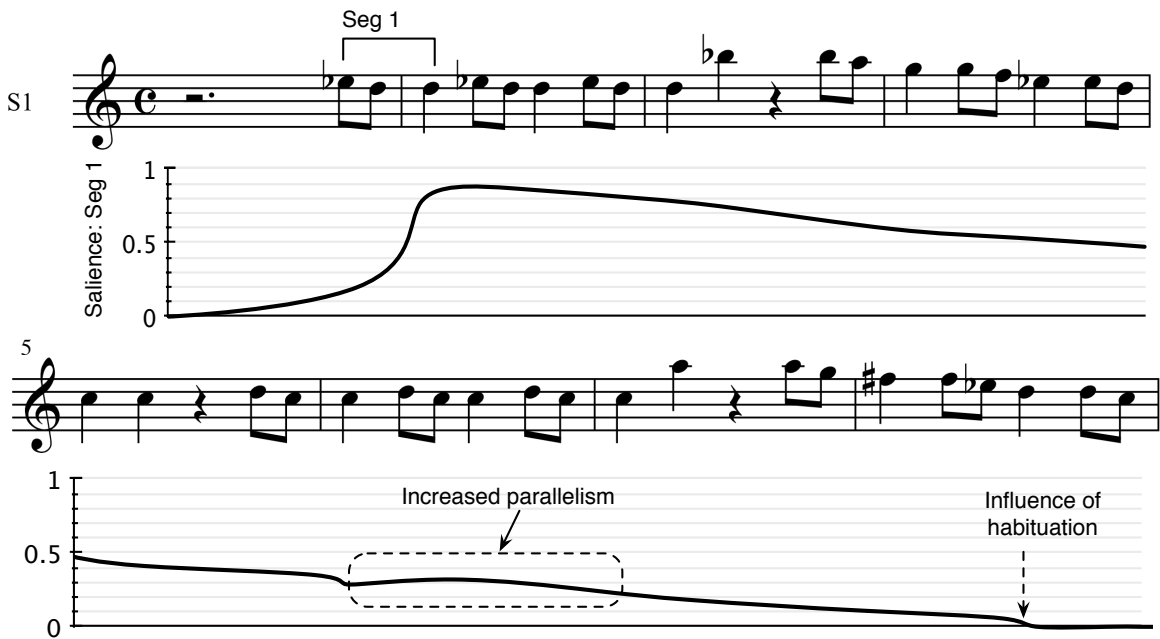


Figure 5.8: Evolution of the cognitive salience over time for the opening segment (“Seg 1”) in Mozart’s 40th Symphony. The rate of decay is slowed in m. 7 by the imitative restatement of segment 1, transposed down one scale-step.

In Figure 5.9 we see the development of WM capacity over the course of training on the Bach BWV 1013, Allemande (top) and the solo flute work *Invidere* (bottom). Each work was run through repeated training iterations, until no further nodes, edges, or links were added to the CM. The dotted line in each plot indicates the number of elements (segments and/or chunks) in WM and the solid line indicates the number of individual events contained within those segments/chunks. On both plots we see that the WM contents tend to stabilize into a somewhat fixed pattern as a result of the gradual replication of WM segment/chunk structure in LTM. We also see that the contemporary work, which has a much less regular/symmetrical formal structure, produces greater variance of WM contents than

the Bach, which produces a fairly consistent ratio of events to elements. The higher *maximum* event capacity of *Invidere* (Maxwell = 130, Bach = 104) is likely due to the fact that this work contains several passages containing rapid note repetitions, which are easily chunked, allowing a greater number of events to be retained in WM. On the other hand, the higher degree of parallelism in the Bach results in a higher *average* number of elements (segments/chunks) retained in WM (Bach = 16.85, Maxwell = 12.14), and also a higher average number of individual events (Bach = 96.59, Maxwell = 56.99), likely due to increased cognitive salience. Thus the Bach, having a more “chunkable” structure, is able to compress more elements in WM than the generally less “chunkable” Maxwell.

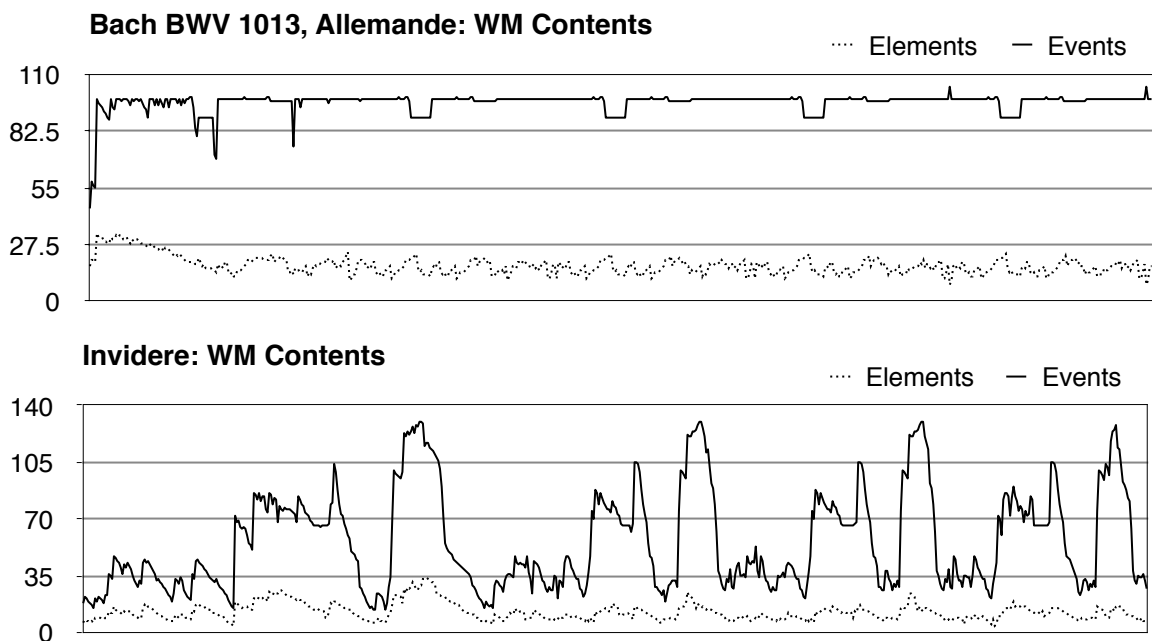


Figure 5.9: Evolution of WM capacity during training on repeated iterations, showing the number of elements (segments and/or chunks) retained and the number of individual events contained by those elements for the first Bach partita (top) and the Maxwell work, *Invidere* (bottom).

5.2.2 WM Chunking and Higher-level Segmentation

Looking now at the chunking process, Figures 5.10 and 5.11 show the chunking structure when trained on the Mozart melody, after the first and last training iterations, respectively. In this diagram, the event-level stream is recorded in the top staff, and each staff below records the boundary events at a higher level of form. The information is extracted from the

stream during training, and the data is updated after each iteration of the UPDATEWM algorithm (Algorithm 4.6) run during CM training (Section 4.2.3). Note that in the first iteration, MusiCog does not retain the opening segment—as the number of individual, unchunked events exceeds the WM capacity—and also that an incorrect higher-level boundary is detected at the end of m. 4. It is worth pointing out that such “false positives” will be retained in LTM, once learned, since the CM currently has no mechanism for *forgetting*. However, both of these errors are corrected in the final training iteration (Figure 5.11), which shows a concise and musically intuitive hierarchical structure.



Figure 5.10: Formation of chunk structure in WM after the first training iteration on the melody from Mozart’s 40th Symphony.

Figures 5.12 and 5.13 show a similar chunking representation of the first and last training iteration on the Bach BWV 846 Fugue. Here we see that the more complex material produces a more fragmented set of chunks in the first iteration, but that completed training once again produces a concise hierarchical form. Figure 5.13 also highlights the process through which higher-level boundaries are created by the discovery of parallelisms between new segments and the current contents of the WM. The phrase boundary in m. 2 is

The image displays a musical score for three staves, labeled S1: L1, S1: L2, and S1: L3. The music is in C major, 4/4 time. The first system (measures 1-4) shows a melodic line in S1: L1 with eighth-note patterns, a bass line in S1: L2 with quarter notes, and a tenor line in S1: L3 with half notes. The second system (measures 5-8) continues the melody with a key signature change to C minor in measure 7. Arrows labeled 'Low-level boundaries' point to specific notes in the S1: L1 staff.

Figure 5.11: Formation of chunk structure in WM after training on Mozart's 40th symphony is complete.

discovered when the opening motive is repeated at the dominant pitch level (G, A, B, C), outlined in grey in the example. It is also worth noting the altered boundary structure at the beginning of m. 7 (outlined), where the strong sense of closure produced by the preceding segment causes MusiCog to place the new boundary one note prior to the restatement of the opening motive. The rhythmic augmentation following the series of eighth-notes (C, F, D), to the quarter-note on G, does suggest a boundary, but tracing the segmentation process in detail, we also noted that the E at the start of m. 7 is actually detected as a singleton segment (see Section 4.2.1, p. 86). In order to resolve the singleton, the E is shifted to the new segment starting on C due to the closer rhythmic proximity of the C than the preceding G (i.e., eighth-note versus quarter-note).

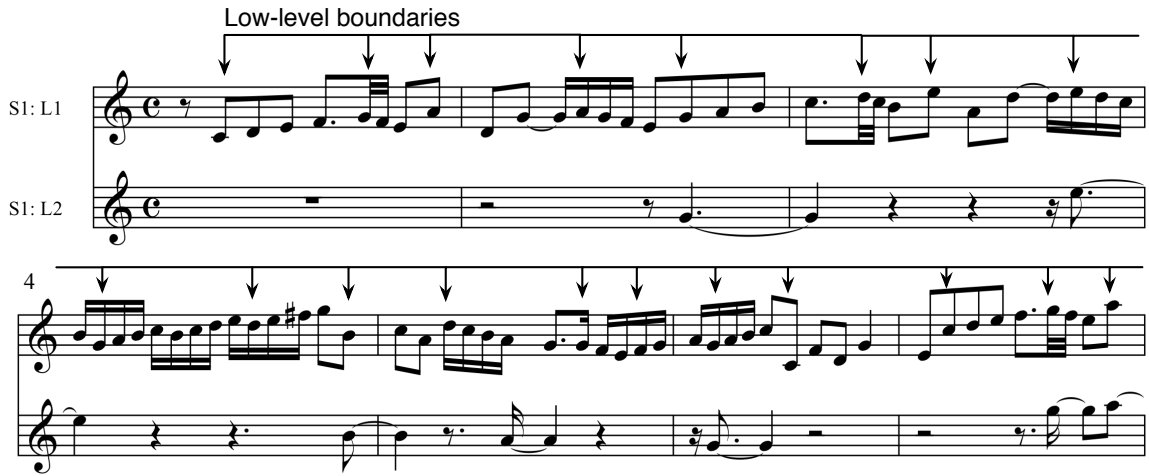


Figure 5.12: Low-level segmentation of the theme from Bach's BWV 846 Fugue showing the state of hierarchical learning after the first training pass.

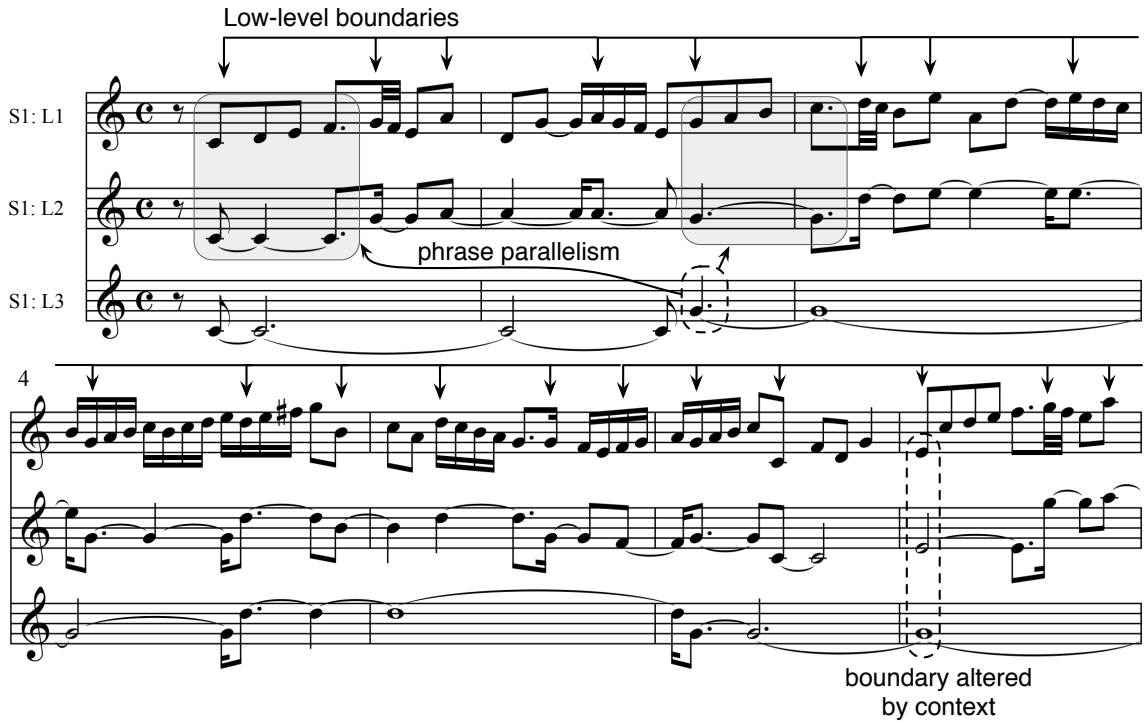


Figure 5.13: Low-level segmentation of the Bach theme after training is completed.

5.3 Long-Term Memory: Learning in the CM

Learning in the CM creates a complex data structure that is difficult to visualize in a concise manner. However, one way to get a general sense of the structure being learned is to look at how the model acquires new information during training by plotting the growth of the CM graph through time. In order to track the development of the CM graph, we trained the model sequentially, learning the complete representation of each work before moving on to the next. In Figure 5.14 we see the development of the pitch and rhythm models when trained consecutively on each of the four movements of Bach's A Minor Partita, BWV 1013. Each movement was presented iteratively until no further nodes, edges, or links could be added to the CM graph.

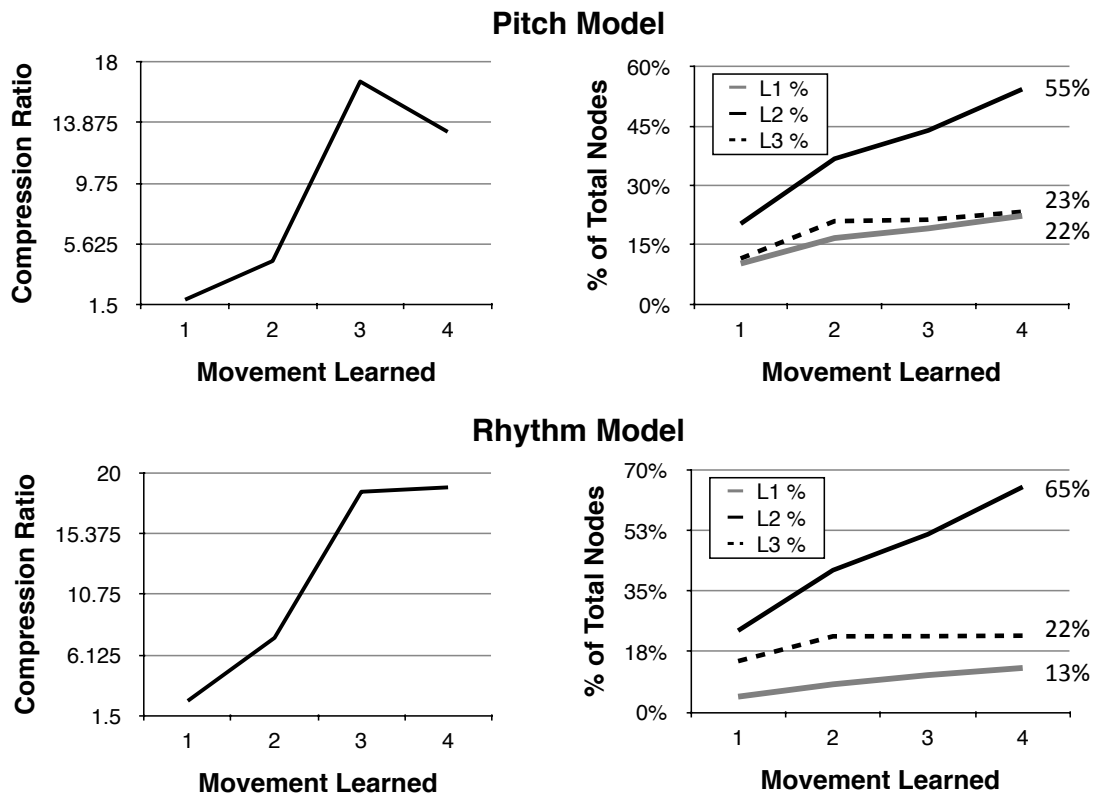


Figure 5.14: The development of the pitch and rhythm models when trained consecutively on the four movements of Bach's A Minor Partita, BWV 1013. The two left-hand charts indicate model compression and the charts on the right indicate the percentage of nodes learned at L_1 , L_2 , and L_3 .

The upper charts represent learning in the pitch model and the lower charts represent learning in the rhythm model. The two left-hand charts indicate model compression, measured as the number of events in each movement divided by the number of nodes added. The number of events is calculated as the number of inputs $\times 3$, in order to account for the Schema, Invariance, and Identity information associated with each event. The charts on the right indicate the percentage of nodes learned at L_1 , L_2 , and L_3 , measured against the total number of nodes added. Note that the pitch model achieves an average compression ratio of 9.04, with a clear trend of increasing compression, and the rhythm model achieves a higher overall compression ratio of 11.88. The difference can be understood by looking at the charts on the right. Here, we see that the rhythm model shows a dramatically lower portion of learning at L_1 , suggesting a general lack of rhythmic variety at the motive level. This is also clear from studying the score², which shows a predominance of isochronous sixteenth- and/or eighth-note patterns. As was mentioned in Section 4.2.1 (p. 85), this kind of rhythmic uniformity tends to exaggerate the influence of pitch information in low-level boundary detection, causing the L_2 structure of the rhythm model to mimic that of the pitch model (i.e., since the rhythmic material alone has little impact on segmentation). This behaviour is reflected in the similar portions of higher-level (L_2 and L_3) learning observed across the rhythm and pitch models, suggesting that formal structure in the Bach is focused not at the motive level, which is dominated by fairly generic, ornamental gestures, but rather at the phrase level. Figure 5.15 shows the node counts at each tier of the pitch model, when trained on the Bach Partita.

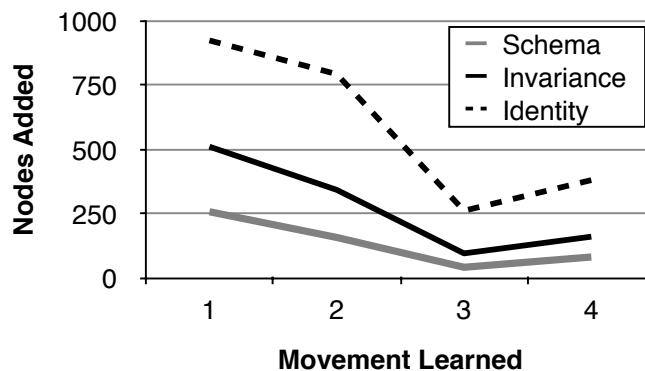


Figure 5.15: Nodes added at the Schema, Invariance, and Identity tiers (pitch model) when trained on Bach’s A Minor Sonatas and Partita, BWV 1013.

²A publicly available transcription can be found at <http://www.mutopiaproject.org/ftp/BachJS/BWV1013/bwv1013/bwv1013-let.pdf>

Note that the proportions of nodes learned at each tier reflect the compressibility of the three representations—i.e., Schema values, or contours, use only three symbols (+, -, 0), the Invariance representation is an unsigned interval, thus representing approximately half of the total intervals, and the pitches themselves represent all events.

In Figure 5.16 we see a similar plotting of MusiCog’s learning over time, this time trained on a corpus of Finnish folk songs. Of note here are the generally higher mean compression ratios—pitch = 52.99, rhythm = 260.99—due to the simpler, more symmetrical structure, and overall shorter duration of the folk songs, relative to the Bach. It will be noted that reduced compression, associated with a sharp increase in learning—particularly at L_2 —occurs at song 12 (circled). This suggests that a considerable degree phrase-level novelty was introduced to the corpus in this particular work. Of note also is the fact that rhythmic L_1 learning is relatively flat at this point, whereas learning at L_2 increases dramatically (a similar, though less pronounced effect is seen in the pitch model). This kind of dramatic increase in learning can occur as a result of new Schema tier patterns being learned, which require subsequent completion at the Invariance and Identity tiers.

The general tendency to learn more structure at L_2 than L_1 indicates an important feature of both the Bach Partita and the Finnish folk songs; i.e., that variety is achieved primarily through phrase-level structure. This tendency can also be seen in Jazz music, as shown in Figure 5.17, where we again see an emphasis of L_2 learning over L_1 , when trained on a corpus of jazz songs (pitch and rhythm are averaged in this chart). This emphasis on phrase-level novelty underlines an important principle of Cope’s notion of music recombinance; i.e., that new music is a recombination of existing musical materials. Thus, a relatively small set of motivic segments can yield a relatively large corpus of novel phrases. While this is a somewhat obvious corollary of the combinatorial nature of music³, it is nevertheless an important aspect of musical form that is inaccessible to many music learning algorithms. By using a multi-level cognitive approach, MusiCog is able to differentiate between segment-level and phrase-level structure, and thus to build a memory model that reflects this basic, cognitively grounded, creative strategy of human composers.

It is important to note that low-level segmentation has a strong effect on hierarchical learning, and that this effect does not follow a simple linear pattern. Although increasing the value of φ will generally lead to the creation of longer L_1 segments, this may consequently impede the phrase boundary detection mechanism (Section 4.2.2), negatively impacting

³By analogy to natural language we can consider comparing the number of novel words to the number of novel sentences—“words” being encoded at L_1 and “sentences” encoded at L_2 . Clearly there are a staggering number of novel sentences that can be created with even a relatively small lexicon.

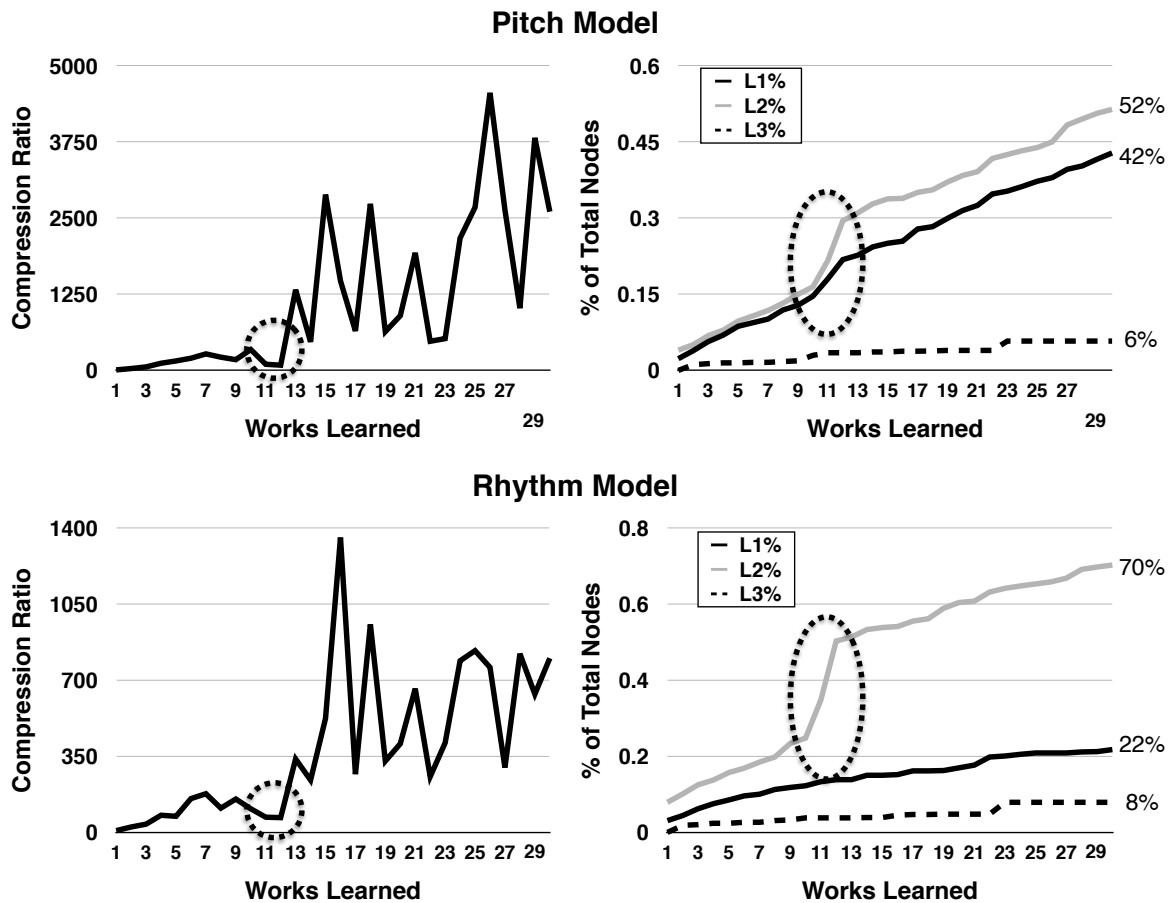


Figure 5.16: Learning in the pitch and rhythm models when trained on the Finnish folk song corpus ($\varphi = 0$). Of note is the dramatic increase in learning at L_2 for song 12 (circled), suggesting that this particular song introduced fairly significant phrase-level novelty.

MusiCog’s ability to learn structure above L_1 . A hypothetical example is shown in Figure 5.18, where we see a clear, non-contiguous segment parallelism (C,D,E) exploited to create two chunks in 5.18a. However, when the φ value is increased in 5.18b, the altered low-level segmentation hides the parallelism within the second segment, impairing MusiCog’s ability to detect the higher-level form. Thus the selection of the φ value has a direct impact on the learning of hierarchical structure, and the capacity to learn such structure is dependent upon the musical content itself.

Since training on consecutive iterations of a single work represents a somewhat unrealistic situation, it is generally preferable to train MusiCog by presenting works in a randomized order. When the order is randomized, similarities between different works can be detected by the CM, potentially resulting in a more compact CM graph structure for the

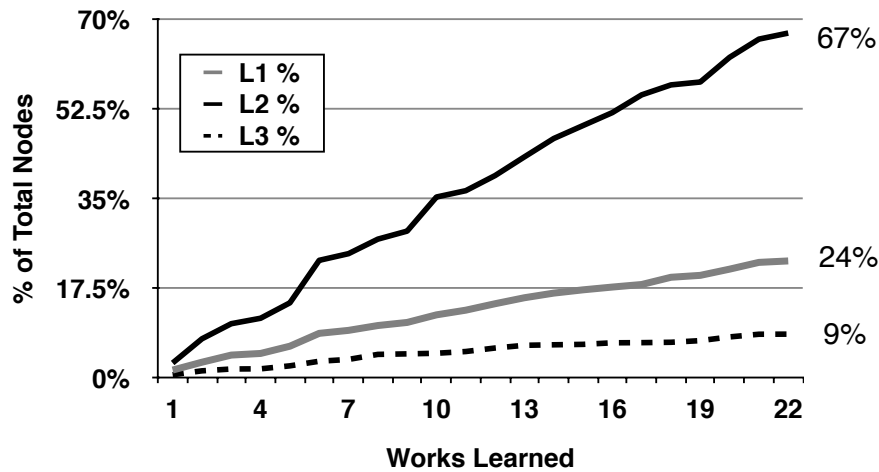


Figure 5.17: The distribution of nodes learned at levels 1, 2, and 3 of both the pitch and rhythm models, when trained on a corpus of jazz songs.

entire corpus. Table 5.1 shows the average numbers of nodes learned at L_1 , L_2 , and L_3 for four different φ values, when trained on a Jazz corpus (top), and the four movements of Bach’s BWV 1013 (bottom), over four complete training attempts (i.e., four randomized orders of presentation).

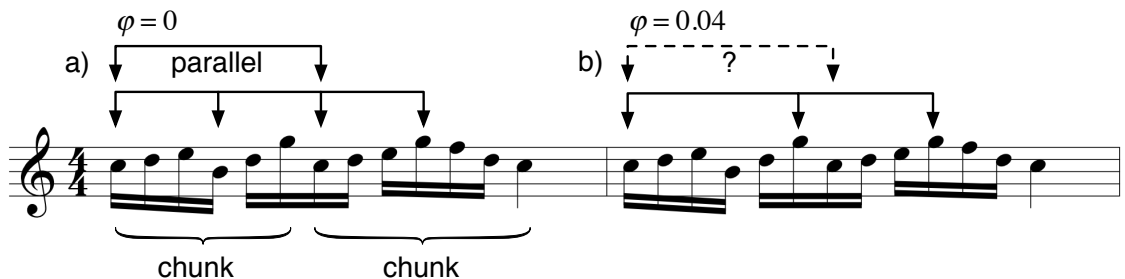


Figure 5.18: By altering low-level segment structure, higher φ values can negatively impact the detection of higher-level parallelisms.

Since the general goal of the CM’s learning algorithm is to find the most compact hierarchical representation for the learned corpus, it seems reasonable to suggest that reduced node creation (i.e., maximal compression) might be a good criterion for determining the φ setting. However, it is also important to consider variance in the total number of nodes created, across different random orders, since high variance suggests that the segment structure is particularly sensitive to the order of presentation. Looking at Table 5.1, when training on the Jazz corpus, $\varphi = 0.025$ creates the fewest nodes, but also has the highest standard deviation of node counts across the four attempts. The total node count for

Mean Nodes (4 attempts): Jazz Corpus					
φ	L_1 nodes	L_2 nodes	L_3 nodes	Total Nodes	Std Dev
0.0	1201.25	2064	517	3823.25	35.23
0.01	1188	2114.75	506	3785.75	34.98
0.025	1207.25	2083.25	471	3761.5	54.13
0.04	1212	2164.25	462.5	3838.75	38.35

Mean Nodes (4 attempts): Bach BWV 1013					
φ	L_1 nodes	L_2 nodes	L_3 nodes	Total Nodes	Std Dev
0.0	882.75	2008.75	562.5	3454	38.82
0.01	917.5	2001	513.25	3431.75	27.5
0.025	1055.25	2066.75	459.25	3581.25	63.06
0.04	1107.25	1963.5	390.5	3461.25	129.60

Table 5.1: Learning at L_1 , L_2 , and L_3 for four different values of φ when trained on a corpus of jazz songs (top), and on the Bach BWV 1013 partita (bottom).

$\varphi = 0.01$ is slightly higher (approx. 1%), but with significantly lower standard deviation, so we consider this to be a more appropriate φ setting for this corpus. For the Bach corpus, we see that $\varphi = 0.01$ creates the fewest nodes, and is also the least sensitive to order of presentation (i.e., it has lowest standard deviation), suggesting that it will also provide an appropriate setting for this material⁴.

5.4 Generation in the PM

As mentioned at the beginning of Chapter 4, testing of MusiCog’s generative capabilities focused on the notion of musical style imitation. During training, we used a randomized training process that avoided consecutive presentations of a given source work, with a stopping condition defined as the point at which MusiCog stops adding new nodes, edges, or links to its CM graph (i.e., no new structure is learned in LTM; see Section 4.2.3). The duration for each generated work was chosen to approximate the average duration of works in the training corpus. Generation began with top-down planning, after which bottom-up prediction was used to construct new plans, in the event that the initial plan failed to generate the desired duration of music (see Section 4.2.4). Because MusiCog generates in segments, the precise PM duration of a given generation varied according to the

⁴Although this may seem to suggest that $\varphi = 0.01$ is a generally optimal setting, it is worth noting that training on the Folk corpus used for generation testing in Section 5.4 revealed an optimal setting of $\varphi = 0$.

generated segment structure. To provide a degree of closure, each generated melody ended with the boundary event following the last generated segment. The option for MusiCog to learn from feedback was disabled, except where explicitly indicated. Supporting materials for the following discussion of MusiCog generation can be found at:

<http://www.sfu.ca/~jbmaxwel/MusiCog/index.html>

At this site you can find audio and score examples of MusiCog's melodic generation, downloads for the MusiCog framework (Objective-C, 64-bit), the Max external and help file (Max 6.1 or later, 64-bit), and MIDI files for all training corpora and MusiCog generations.

We used two analysis packages during testing: 1) Eerola and Toiviainen's "MIDI Toolbox" for MATLAB [78], and 2) the SIMILE melodic analysis package by Müllensiefen and Frieler [173]. From the MIDI Toolbox we focused on two analysis metrics: entropy and expectancy-based complexity. The `entropy` function calculates the entropy of pitch/interval transitions in the analyzed work. The expectancy-based complexity measure (`complexbm` in the Toolbox) is based on Eerola and North's model [77], and takes into account tonal, intervallic, and rhythmic factors. The tonal factors are based on Krumhansl and Kessler's "tonal stability" measure [125]—a measure of the relative perceptual prominence of pitches in tonal music—and are also influenced by metrical position and duration, both of which contribute to the perceived salience of pitches in melodic contexts. Intervallic factors are based on Narmour's I-R model of melodic expectancy, and rhythmic factors include considerations for rhythmic variability, the degree of syncopation, and the density of the rhythmic passage (i.e., number of events per second). From the SIMILE package we used the `opti3` measure; a multi-factor measure designed to provide a cognitively-grounded estimation of melodic similarity. `opti3` is based on a weighted linear combination of three component distance measures from the SIMILE package: `nGrUkkon` (an n-gram based, pitch interval measure), `rhytFuzz` (a fuzzy estimation of rhythmic similarity), and `harmCorE` (a correlation-based measure of implied harmonic structure). This combination of measures was designed specifically for the analysis of similarity in large corpora, and was optimized to fit the similarity judgements of human experts [172]. A detailed description can be found in the SIMILE documentation [173].

To aide visualization of the different corpora, and provide an intuitive baseline for understanding musical differences between test melodies, we also included a set of randomly generated melodies. For each test case the number of random melodies was equal to the

number of generated melodies (which was equal to the number of corpus melodies). Random melodies were created by randomly selecting MIDI pitches in the range [48,96] (C2 to C7), and randomly selecting rhythmic values from the set of defaults used during rhythmic quantization (see Section 4.1). An excerpt of a “random” melody is shown in Figure 5.19.

Figure 5.19: Excerpt of a “random” generated melody.

Permutation testing

To test for significance in our complexity and entropy measures we used a permutation testing approach, with correction for multiple comparisons, as implemented by Groppe [95] (after Blair and Karniski [23]). Permutation testing, also known as randomization testing, involves the comparison of an initial distribution—the *test statistic* T_0 —to a set of random permutations of that distribution, referred to as the *reference distribution*. The method seeks to estimate the correct distribution of the test statistic under a null hypothesis. The underlying assumption is that, if the null hypothesis is true, then all possible permutations of observations should be equally likely to occur [143]. The method is well suited to musical data, as it is applicable to small samples, and removes the need for *a priori* knowledge about the statistical population (e.g., the assumption of normality in the distribution of “all” folk songs). Determining significance involves calculating the ratio between 1) the number of permutations of the reference distribution with an absolute *t*-value $|t^*|$ greater than or equal to that of T_0 (i.e., $|t^*| \geq |t|$), and 2) the total number of permutations run. If a large

number of permutations have a $|t^*| \geq |t|$, we can assume that t lies near the middle of the reference distribution, and that the groups being tested represent a random selection. If this is the case, we can safely accept the null hypothesis H_0 . On the other hand, if very few permutations produce a $|t^*| \geq |t|$, then t must lie near the tail of the reference distribution. In this case it is reasonable to assume that there's something particular about the ordering of the original data (e.g., the collection of test melodies), such that permutation of its elements alters it in a fundamental way. Since encountering such a specifically ordered distribution of elements is improbable under H_0 , we can safely reject the null hypothesis.

Because the measurements provided by SIMILE are estimates of the melodic *similarity* between a given melody and the corpus—i.e., the data point for each melody represents its distance from all other melodies—all observations involving a single melody are correlated and we cannot assume independence. For this reason we use an alternative method developed by Gonzalez Thomas et al. ([93] in publication), designed to remove need for independence of observations, when testing our `opti3` results. This approach reorganizes the results data such that T_0 accounts for both within-group and between-group differences, and adapts the t -value calculation accordingly. First, a matrix of similarity ratings $s(x,y)$ is created, like that shown in Figure 5.20.

	A	B
A	$s(A,A)$	$s(A,B)$
B	$s(B,A)$	$s(B,B)$

Figure 5.20: Matrix of similarity ratings by Gonzalez Thomas et al.

Here, each quadrant represents the similarity ratings for pairs (A,A) , (A,B) , (B,A) , and (B,B) . The t calculation estimates the mean difference both within and between groups, using Equation 5.1.

$$t = (\overline{AA} + \overline{BB}) - (\overline{AB} + \overline{BA}) \tag{5.1}$$

where \overline{AA} , \overline{BB} , \overline{AB} , and \overline{BA} are the means of similarity readings for each quadrant. As when using Groppe's function, our test statistic T_0 is the set of similarity readings from test groups (A,B) , and t is calculated using Equation 5.1. Permutation randomly permutes the column labels, so that the position of each melody in the set of groups is randomly varied,

potentially changing the melody’s group membership. As with the complexity and entropy tests, we run 10,000 permutations for each pair, with an alpha of 0.05⁵.

5.4.1 Testing on the Folk Corpus

We began by training MusiCog on a corpus of 100 folk songs compiled by Gonzalez Thomas et al. [92], from which 100 16-measure melodies were generated. Recognizing that cohesion tolerance (φ , see Section 4.2.1, p. 86) has a direct influence on segmentation, and thus potentially on the recombinant aspects of generation, we ran the training with two different settings: $\varphi = 0$ and $\varphi = 0.05$. Figure 5.21 plots complexity and entropy ratings for the folk, random, and MusiCog groups, at each cohesion tolerance setting. Here we see a clear overlap between the MusiCog generations and the training corpus, and a spatially distinct cluster for the random melodies. It will also be noted that increasing the φ value resulted in higher complexity ratings ($\mu = 4.49$ to $\mu = 4.63$), and slightly increased variability in the entropy ratings (from standard deviation $\sigma = 0.003$ to $\sigma = 0.0032$).

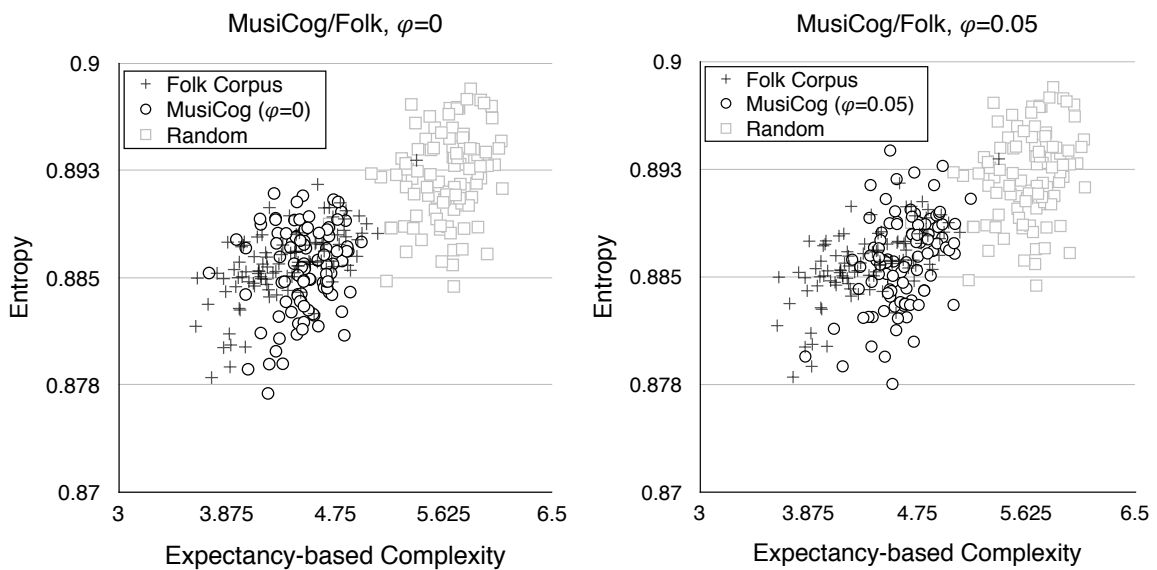


Figure 5.21: Comparison of entropy and expectancy-based complexity ratings for MusiCog’s generated folk melodies, a folk melody training corpus, and a set of “random” melodies.

⁵The alpha indicates the significance level of a test, and is the probability of making a Type I error (i.e., a so-called “false positive”).

These increases are likely due to the decreased segmentation sensitivity caused by the higher φ value, which tends to promote the creation of longer segments in the PE, and potentially less compact hierarchical representations in the CM (i.e., more nodes added for a given corpus—see Section 5.3, p. 141). For example, with the folk corpus, we observed that training with $\varphi = 0$ resulted in the creation of an average of 20 segments per melody, whereas training with $\varphi = 0.05$ produced an average of only 15 segments per melody. As segments become longer, they capture more intra-opus information, becoming more specific to a given source work, and consequently less amenable to recombination (an implicitly inter-opus phenomenon). These longer segments generate their own internal expectancies, which may be quite unique in the corpus. When recombined by MusiCog, such strong internal expectancies are more likely to be violated by the transitions *between* segments, potentially leading to increased formal complexity; musically speaking, to an impression that the formal goals of the melody are changing with each new segment. By contrast, shorter segments are more generic and thus more amenable to recombination, since a larger portion of the expectancies generated tend to be built *across* segments, thus capturing more inter-opus information.

Significance testing for complexity and entropy revealed a significant difference for complexity, but no significant difference for entropy ($p = 0.539$). Turning to the SIMILE tests, we ran the `opti3` similarity measure on all three groups (folk, MusiCog, and random), visualizing the results using classical multi-dimensional scaling, as seen in Figure 5.22. Once again we see a distinct cluster formed by the random group, and a notable overlap between the MusiCog group and the folk corpus. However, testing the results using the method of Gonzalez Thomas and et al. revealed a significant difference between the MusiCog group and the folk corpus⁶, suggesting that MusiCog had not replicated the folk style adequately across the total collection of generated melodies.

In order to gauge the effect of feedback learning on generation, we retrained MusiCog on the folk corpus (with $\varphi = 0$), and generated 1000 example melodies with feedback learning enabled. We then tested complexity and entropy ratings for the non-feedback melodies against the last 100 feedback melodies (i.e., melodies 901-1000) and the folk corpus (with expected means taken from the folk corpus). As in the non-feedback case, all measures for feedback melodies 901-1000 produced significantly different complexity ratings, but in this case a weakly significant ($p = 0.035$) entropy rating was also observed. More crucially, significant differences were also found between the feedback melodies and the original,

⁶Given the MDS plot, the random group was assumed to be significantly different.

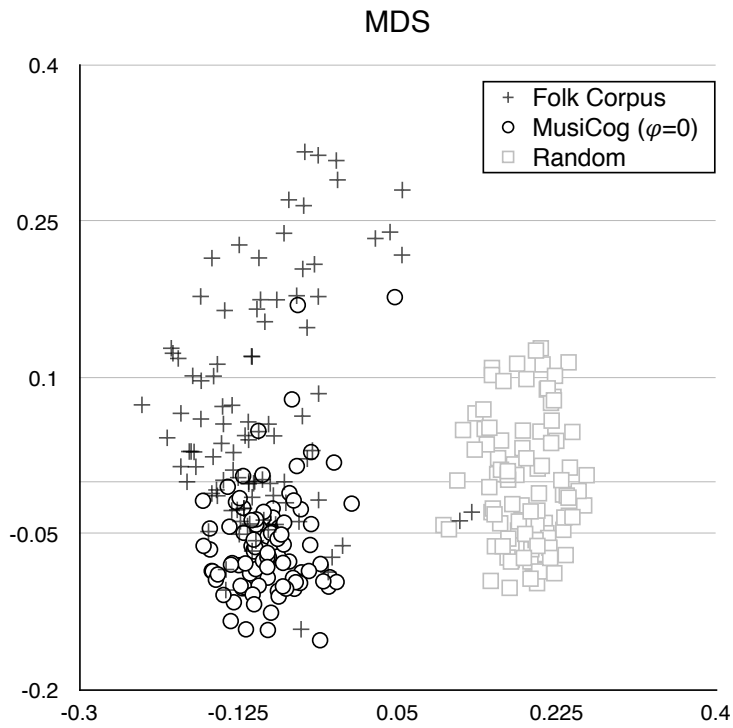


Figure 5.22: Multi-dimension scaling plot of the Folk corpus, the MusiCog generations, and the set of random melodies.

non-feedback melodies, as can be seen in Figure 5.23, where we see a noticeable effect of feedback learning on ratings for both dimensions. Of course, given that MusiCog’s generated melodies tend to exceed the complexity and entropy of the folk corpus, it is intuitively clear that learning from its own output—and thus being *further influenced* by its own tendencies—would likely exaggerate this behaviour. It is also worth noting that the CM graph size increased considerably with feedback learning enabled.

Score Analysis of MusiCog Generation

Because results from the statistical tests do not provide enough information to understand the musical differences between MusiCog’s melodies and the folk corpus, we turned to the musical scores for further analysis. Two sample scores were selected; the folk melody in Figure 5.24, and MusiCog generation 43, shown in Figure 5.25. Looking at the folk song, we see a clear binary structure, expressed through a symmetrical, two-part “call-response” form. The two main themes, **A** and **B**, are four bars long, and each is constructed from a

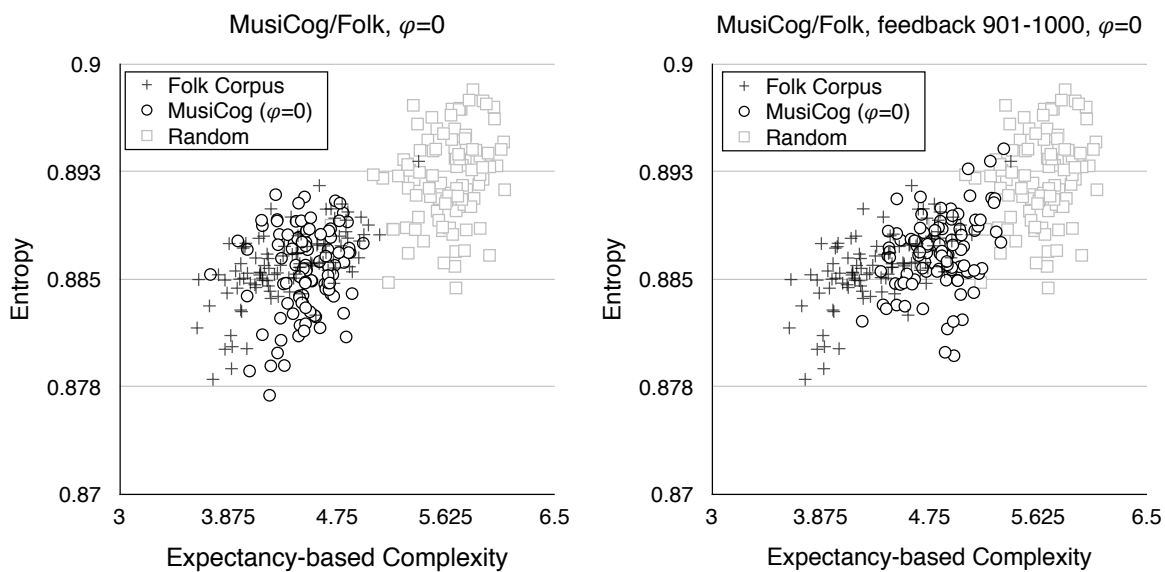


Figure 5.23: Repeated generation (1000 melodies) with feedback learning enabled produces a clear and significant increase in the complexity and entropy ratings of the generated melodies.

pair of contrasting two bar phrases. The melody expresses a clear tonality, beginning on G and ending on the tonic of the key, C. Themes **A** and **B** likewise emphasize the dominant and tonic, with the second bar of phrase **A**_i ending on G, and the second bar of **A**_{ii} ending on C. Theme **B** follows the same formal plan.

Turning to MusiCog generation 43, shown in Figure 5.25, we see that it is composed entirely in the key of C major. Recalling the operation of the mode/tonality induction function (Section 4.2.1), and its application in the PM (Section 4.2.4), we can see that, by the end of the second measure, the PE could quite easily induce the key of C major (i.e., Ionian mode, salient pitch-class C), due to the durational emphasis of C5⁷ and the lack of any ambiguating accidentals (e.g., the absence of F \sharp , which might support G major). From this point forward, MusiCog will have “decided” on the scale for the generation, and pitch content will be quantized to the induced scale⁸.

⁷In these two measures C5 occupies 2.5 beats, A4 occupies 1.5 beats, and all other pitches occupy only 1 beat.

⁸This will remain the case until a sufficiently chromatic segment is generated from LTM, in which case the segment will be left unquantized. However, the established mode of C Ionian will remain in force, in the event that quantization is required again later in the melody.

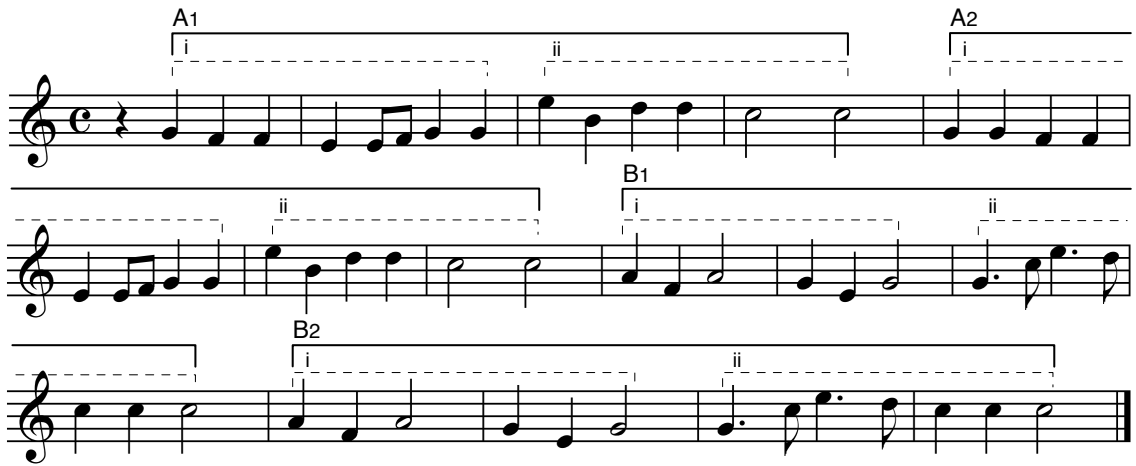


Figure 5.24: Folk song at median complexity from training corpus.



Figure 5.25: MusiCog folk generation number 43.

Signs of Tonality

MusiCog currently lacks an explicit model of tonality, and thus had no deliberative routine to structure melody 43 in a way that would promote a strong tonic function for C. For example, whereas the folk melody aligns cadences with strong metrical accents, placing the tonic on the downbeats of mm. 4, 8, 12, and 16, the “tonic” in MusiCog’s generation is accented much less symmetrically at mm. 5, 7, 8, and 9. Perhaps most crucially, C does not appear at all in the last five measures of the melody⁹. Nevertheless, MusiCog’s generation does contain transitions with strong tonal implications, and these transitions do at times align with an implied metrical structure. For example, the B4 at the end of m. 4 implies a strong cadential relationship with the consequent C5 (downbeat of m. 5), and a similar pattern is repeated in mm. 8/9 (though in this case the B4 occurs in a weaker metrical position). It is also frequently the case that non-root chord tones of the tonic and/or dominant appear in accented positions. Recognizing the importance of such tonal implications, we decided to perform a manual harmonic analysis, shown in Figure 5.26, outlining the harmonic structure implied by the melody. Of course, this outline displays a degree of subjective interpretation (the bracketed chords represent choices which, though perhaps less obvious, are entirely idiomatic), but it nevertheless demonstrates the fact that certain tonal cues are present in the generated melody.

The figure shows a musical score for a melody in C major, consisting of 16 measures. The melody is written on a single staff in treble clef. Below the staff, a series of chords are indicated with horizontal lines and labels: I, V, I, (vi), V, I, v7, I, (ii), V, I, v7, (I₄⁶), v7. The chords are placed under the notes they harmonize, with some chords spanning multiple measures. The final measure (m. 16) ends with a whole note on C5, which is the tonic.

Figure 5.26: The manual inclusion of a harmonic outline highlights the implied tonal structure of the MusiCog melody.

⁹Emphasizing this point, it is worth noting that replacing the final bar with a whole-note on C5 gives the melody a strong sense of tonal resolution.

In order to determine whether the implicit tonal structure of MusiCog generation 43 was simply an accident, encountered only in this particular melody, we decided to test specifically for indicators of tonality. To this end we used the Toolbox's `maxkkcc` measure, which provides an estimate of the most probable tonality implied by a set of pitches, using Krumhansl and Schmuckler's well known key-finding algorithm [126]. The method exploits Krumhansl and Kessler's 24 key profiles, an example of which (for C major) is shown in Figure 5.27.

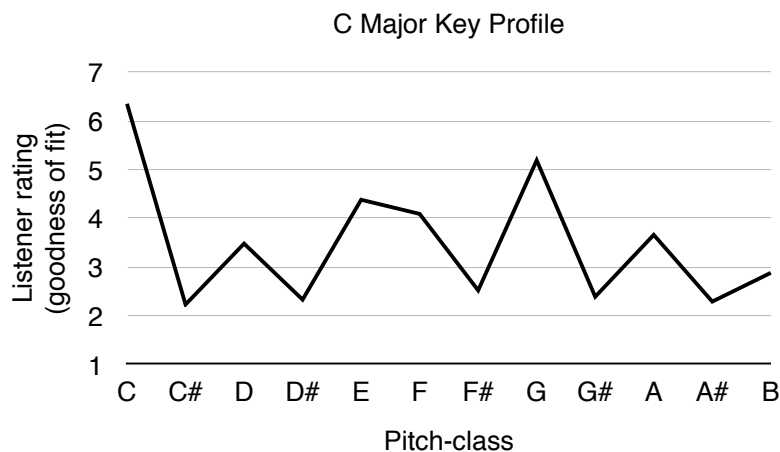


Figure 5.27: The Krumhansl-Kessler key profile for C major. The scale represents listener ratings of goodness of fit for each of the chromatic tones when preceded by either an individual tone or a diatonic chord.

The profiles were derived experimentally via probe-tone tests on human subjects, and provide an estimate of the perceived key membership of each of the 12 pitch-classes, when heard in the context of a specific pitch or diatonic chord [125]. The `maxkkcc` function returns the maximum cross-correlation for all 24 key profiles, and is here used to estimate the degree to which a melody would be perceived as tonal by a human listener.

Testing the MusiCog group against the folk corpus did reveal a significant difference in the mean strength of tonality between the two groups. However, this only tells us that MusiCog did not replicate the strength of implied tonality demonstrated across the entire Folk corpus. In order to understand whether the tonal structure observed in MusiCog generation 43 was purely accidental, or perhaps arose as a byproduct of the pitch quantization mechanism, we needed a baseline for measuring the influence of pitch quantization alone. To this end, we generated a new set of “random” melodies, the pitches of which were constrained to the key of C major. We then tested the group of random *diatonic* melodies against the

MusiCog group, revealing a significant difference, with the MusiCog group demonstrating a higher mean tonality rating than the random (diatonic) group. The histograms for all three groups—Folk, MusiCog, and random (diatonic)—are shown in Figure 5.28.

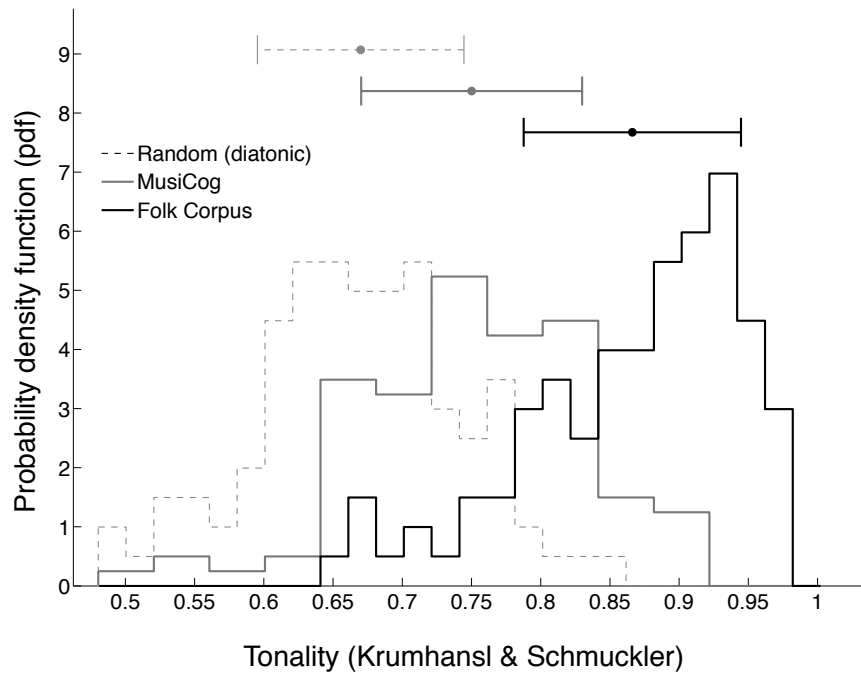


Figure 5.28: Histograms of the strength of implied tonality (using the `maxkkcc` function) of the Folk corpus, the MusiCog generations, and a set of random melodies in C major.

Here we can see that the group of random melodies clearly generates the lowest overall strength of tonality, with a mean of 0.67¹⁰. As might be expected, the Folk corpus generates by far the strongest tonality rating, with a mean of 0.866, while the MusiCog group generates a mean rating between the two groups, of 0.75. Also of note is the fact that the random group’s lower tonality ratings (i.e., Tonality < 0.65) account for a relatively greater portion of its distribution than MusiCog’s low tonality ratings. Conversely, MusiCog’s highest ratings (Tonality > 0.85) account for a relatively higher portion of its distribution than the random group’s highest ratings. In general, the random group’s ratings exhibit a roughly normal distribution, whereas the distributions of the MusiCog group and the Folk corpus are generally skewed toward higher tonality ratings.

¹⁰The above-chance level observed is likely due to the imposed C major scale quantization. Analysis of the unquantized random melodies, for example, produces a mean rating closer to chance, of 0.58.

Motivic Exploitation

In Figure 5.29 we provide a manual analysis of the segment structure of MusiCog generation 43. The analysis emphasizes contour and boundary relationships, indicating a few prominent features of the melody. Determining the actual segment structure of a given generation is difficult and subject to interpretation, since patterns planned by the PM may not always be realized precisely, and segmentation by the PE may differ from the segment structure intended by the PM (as discussed in Section 4.2.4). In such situations, the state inferred in LTM may be different from that used as the context for the preceding PM generation, leading to complex behaviour. Also, it can occasionally happen that the rhythmic segments produced are shorter than the associated pitch segments (see Section 4.3), in which case the PM truncates the pitch segment, altering the segment structure and thereby altering the inferred context (which in turn alters future generation). When performing manual segment analyses, we begin by searching for direct repetitions, preferably at the phrase level. If repetitions can be found, and they appear to occur at the phrase level (i.e., the repeated segment contains identifiable boundaries), we determine a plausible manual segmentation for the phrase. Having identified a set of repeated segments N , we then search the entire melody for segments with contours matching those in N ¹¹. We then label the extracted segments in alphabetical sequence from the beginning of the melody, identifying related motives using prime symbols (e.g., **a**, **a'**, **a''**, etc.). When a motive is altered specifically by rhythmic variation, we precede its name with a superscript “r” (e.g., **r****a**). It is worth acknowledging here the somewhat counter-intuitive segmentation of the final motive, **r****d'**, in Figure 5.29. Since we know that the PM will place a final boundary at the end of the generation, we can safely assume that the second-to-last segment *excludes* the final note. In this case, we see that the penultimate segment (motive **r****d'**) shares the same contour as **d**, so we identify the segment accordingly. Were this generation to continue, the PE would most certainly identify a boundary on the first of the two F5s in m. 17 (a result of the dramatic increase in IOI), effectively “stealing” the F5 from **r****d'**, and placing it in the following segment.

In this analysis we clearly notice the lack of symmetry, when compared with the highly symmetrical structure of the folk melody in Figure 5.24. In spite of this quite obvious difference, the melody does demonstrate a few important formal properties of its human-composed counterpart. First, we see that there is a clear process of motivic exploitation,

¹¹We search for matching contours because this is the attribute used for identifying and exploiting parallels in the PM.

Figure 5.29: Analysis of the segment structure of MusiCog generation number 43.

arising from the identification of parallelisms in WM, and their compositional exploitation by the PM. This is particularly noticeable in the repetition and variation of motives **a**, **b**, and **c**. These motives undergo transformations of pitch level (i.e., transpositions imposed by the L_2 contour), interval content, and rhythmic structure, creating imitative variations of the original motives. Motive **a'**, for example, is an interval variation of motive **a**, while **r_a** is a rhythmic variation of **a**, realized by halving its rhythmic durations. Motives **r_b** and **r_c** are likewise rhythmic variations of **b** and **c**. Motives **r_{a'}** and **r_a** are truncated versions of **a**, indicating either that the pitch segment has been truncated due to a mismatched rhythm segment, or that the terminal node used to define these segments was located at a lower depth along the same CM path as the terminal for **a**.

Figure 5.29 also draws attention to the multi-segment phrase **2b**, which appears in direct repetition in the melody, at mm. 2/3 and 7/8. Since the PM does not implement any method to simply copy a phrase directly from WM, the generation of such a structure suggests that an L_2 parallelism from LTM has been exploited (i.e., via the mechanism outlined in Section 4.2.4). This motivic pattern also occurs in mm. 5-7, though it is somewhat less obvious in this instance, due to the rhythmic variation of motives **b** and **c**. It is also worth noting that the entire melody can reasonably be divided into two parts. Part **A** develops motives **a**, **b**, and **c**, and runs from mm. 1-9.5. Part **B** introduces motives **d** and **e**, while also continuing to draw frequently on motive **a**, and runs from m. 9.5 to the end. Notable

structural differences in part **B** include its lack of direct repetitions, its wider pitch range, and its introduction of rhythmic syncopation in mm. 14-16, all of which give it a more developmental quality. This added sense of melodic development may offer some explanation for the increased complexity scores found in our quantitative testing discussed in Section 5.4.1.

In contrast to the tendency for motivic *variation* seen in the MusiCog example, the folk song in Figure 5.24 tends to repeat motives and phrases directly, without variation. A possible exception is the pitch imitation between the first two segments of phrase **B1**, which replaces the (-4, +4) interval pattern with (-3, +3). It is worth noting, however, that these two patterns are identical if we use a scale-step representation (i.e., (-2,+2)), rather than an interval representation, in which case it could be argued that pitch imitation is not a formal feature of this particular example melody (such an argument would also then apply to the MusiCog generations). Although the MusiCog melody does repeat material directly, the shifted metrical position of the repeated phrase (i.e., phrase **2b** in mm. 7/8), and its formal separation by a rhythmically varied version (phrase (**a**, **1b**, **1c**) in mm. 5-7), reduce the sense of formal symmetry and balance.

5.4.2 Imitation of more developed styles

In order to better understand the limitations of the “musically naïve” approach to generation currently used in MusiCog, we tested performance with a set of more musically developed styles; a corpus of 22 Jazz melodies, the Bach A minor Partita BWV 1013 for flute solo, and a solo flute work by contemporary composer James B. Maxwell. In identifying these corpora as more “musically developed” we are not making an arbitrary division based on notions of “high” and “low” art, but rather on the level of music theoretical abstraction implicated in their structure. In particular, many of these works follow implicit—and often quite elaborate—harmonic frameworks, which impose direct constraints on the utilized pitch content. In many cases, these harmonic structures are not obvious to the ear, given only the immediate melodic context. This is particularly true of Bach’s music, which contains frequent harmonic shifts and modulations, the logic of which may become clear only after several beats, when the modulatory motion has resolved. Such structures are deliberate expressions of an underlying Pythagorean conception of music (see Section 2), very much *in vogue* in Bach’s time, which sought to reveal music’s deep mathematical structure [128]. In this sense, while clearly guided by listening, Bach’s music is also dependent upon theoretical principles. Which is simply to say that it would not have developed as it did

without theoretical and technological innovation (not the least of which was the invention of a so-called “well-tempered” scale).

In the case of the Jazz corpus we have a slightly different complicating factor, in that all of the melodies have been removed from their original harmonic settings. These melodies, however, would seldom (if ever) be performed without the accompanying harmonic progression, which provides an explicit context for the chromatic violations of key that characterize the style. The Jazz melodies also display an additional layer of ambiguity in their frequent use of rhythmic syncopation. When the overt rhythmic pulse generally provided by the accompanying percussion and “walking bass” is omitted (i.e., when presenting the melody in isolation, as we do here), the highly syncopated melodic writing often pushes the violation of rhythmic expectancy to its limit, dissipating any perceptible pulse. Rhythmic complexity in the Maxwell material presents a similar situation, except that in this case the music is often composed to deliberately *avoid* a sense of pulse (i.e., the pulse is not “lost” through the omission of the pulsed accompaniment, as in the Jazz corpus). This music does not follow a “functional” harmonic plan like the Bach and Jazz corpora, and thus is not specifically relying on unstated harmonic organization, but it also does not remain in a single key, or under a constant scale structure. Thus, the strong impulse for listeners to induce a consistent scale or tonality is deliberately subverted; a situation which, due to the tenacity of our perceptual scale/tonality induction mechanisms, often requires a paradoxically high degree of formal organization to achieve.

Jazz Corpus

We began our testing of more developed musical languages by examining MusiCog’s capacity to generate Jazz melodies. Figure 5.30 shows the expectancy-based complexity and entropy ratings for a corpus of 22 Jazz “standards,” 22 MusiCog generations trained on the Jazz corpus, and 22 random melodies. The requested length for the MusiCog generations was 32 measures. It will be noticed that the Jazz corpus has fairly high variance across both dimensions, perhaps due to the fact that it contains songs from different periods of Jazz, mixing so-called “Big Band Era” songs like “All of Me” with more experimental, avant-garde BeBop songs like “Epistrophe,” by Thelonious Monk. Analysis of complexity and entropy revealed significant differences for both attributes (complexity $p = 0.0285$, entropy $p = 0.0012$), as is apparent in Figure 5.30. However, there is once again a clear (if small) overlap between the Jazz corpus and the MusiCog group, and a distinct cluster for the

random group. Testing using the `opti3` measure from SIMILE likewise reveals significant differences between all groups.

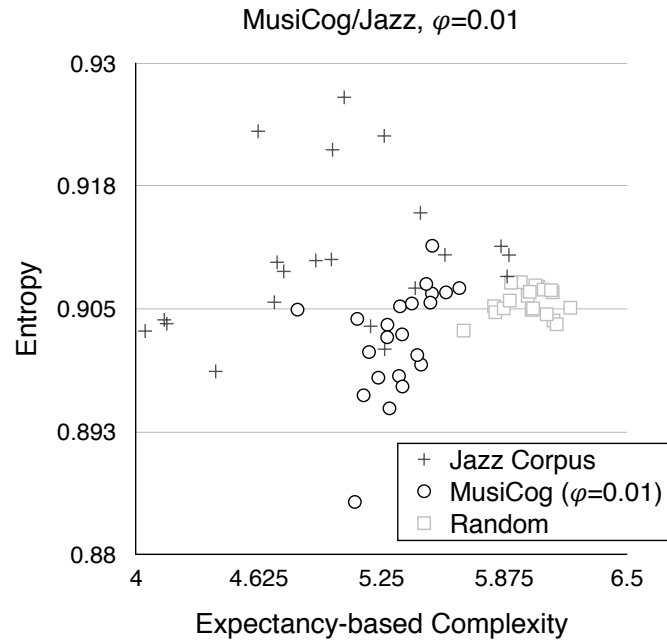


Figure 5.30: Plot of complexity and entropy for a corpus of 22 jazz songs and 22 32-measure ‘songs’ generated by MusiCog.

Figure 5.31 shows the full score for MusiCog’s 18th generation when trained on the Jazz corpus. MusiCog has quantized the pitch scale of the melody to the key of A major. This tonality can be fairly easily established after the first 3 bars, or so, due to the prevalence of pitches from the A major tonic triad (A, C \sharp , E). Figure 5.31 also indicates several places where chromatic passing tones from the minor key (m7, m3, m6) are used, giving the melody an idiomatic Jazz sound. This is facilitated by the use of the PE’s mode/tonality “confidence” rating, which allows the PM to disable pitch quantization in cases where the generated segment suggests a highly chromatic structure. This can be seen in mm. 4, 10, 16, and 17, where the chromatic segments have been surrounded by more distinctly diatonic segments, indicating the temporary disabling of pitch quantization, and subsequent return to the A major scale. This melody has a less convincing tonal structure than MusiCog Folk generation 43, though emphasis of tonic, dominant, and subdominant chord-tones is quite common. There is also an abrupt leap of pitch register at m. 24, which disrupts the melodic continuity somewhat. Rhythmically, we notice the use of “rolled eighth” patterns

throughout¹², giving the rhythms an idiomatic Jazz feel. Thus, in spite of MusiCog’s failure to replicate the corpus quantitatively, important aspects of the style have clearly been captured in some of MusiCog’s generated melodies.

Establish A major

5

9

13

16

20

24

28

Figure 5.31: MusiCog’s 18th generation when trained on the Jazz corpus. Arrows indicate chromatically altered tones.

¹²These rhythms are written using triplets, in order to accurately capture the perceived rhythmic figures. When transcribed for human performers—for example, in Jazz “fake books”—such rhythms are often written using “straight” eighth-notes, with the understanding that the performance practice dictates the use of a “rolled,” or “swing” feel.

Bach A minor Partita, BWV 1013

Next we trained the system on the 4 movements from Bach's A Minor Partita for flute solo, BWV 1013, and generated 4 MusiCog melodies of 112 measures each. Once again, quantitative testing revealed significant differences for all measures; complexity, entropy, and SIMILE's opt_{i3} measure. A multi-dimensional scaling plot of the 4 Bach movements, MusiCog's generations, and 4 112-measure random melodies can be seen in Figure 5.32. It is worth noting that the within-group distances recorded for the Bach corpus are considerably greater than those recorded for the MusiCog or random groups. This is likely due to the fact that, as a set of related movements comprising a larger musical work, the Bach examples could be expected to contain deliberate contrasts, thus demonstrating a higher level of formal/compositional thought. MusiCog, on the other hand, has no conception of multi-movement works, and thus begins each generation with no memory of the work that came before. With no recollection of the preceding generation(s), and no deliberative mechanism for creating inter-opus variety, variation between works can be achieved only through stochastic selection.

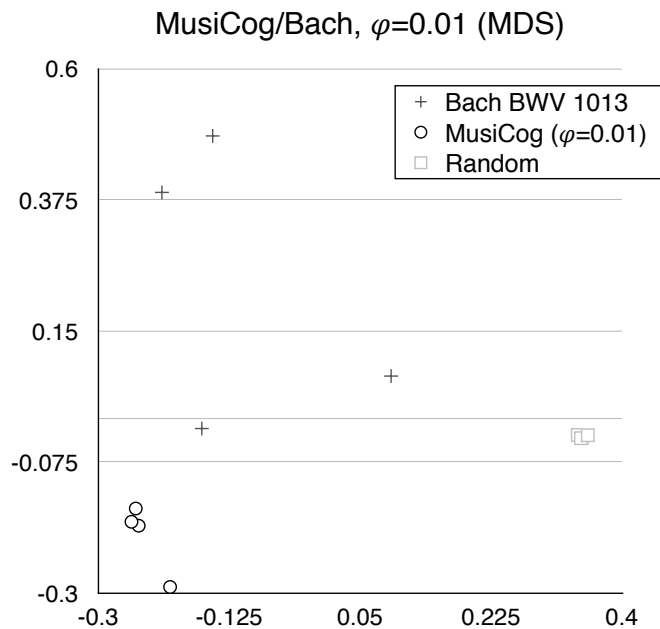


Figure 5.32: Multi-dimensional scaling plot of the 4 movements from Bach's BWV 1013 flute partita, MusiCog's generations, and 4 random melodies.

Of greater interest, however, is the apparent difficulty MusiCog has replicating this particular style. Studying the score, an excerpt of which is shown in Figure 5.33, it is once



Figure 5.33: An excerpt of MusiCog's 3rd generation when trained on Bach's BWV 1013, showing stylistically inconsistent syncopation and rhythmic complexity.

again clear that, although certain aspects of the gestural language of Bach's music are evident, the formal integrity of the whole is clearly lacking. In this case, however, formal problems are apparent both at the phrase level and at the level of the complete work. Again, weaknesses are evident in a general lack of repetition structure and symmetry. However, unlike the previous corpora, the MusiCog Bach generations also introduce an exaggerated and stylistically inconsistent degree of syncopation and rhythmic complexity. In stark contrast to the rhythmic *continuity* displayed by the Bach, the rhythmic surface of all 4 generated works is broken and fragmented, suggesting that there are aspects of Bach's style for which MusiCog's hierarchical, recombinant approach is clearly at a disadvantage. This is a noteworthy outcome, since the Bach works tend to be characterized by extended, isochronous rhythmic patterns (the first movement, for example, is written entirely in sixteenth-notes), which have little or no hierarchical rhythmic structure¹³. Thus,

¹³There is, of course, an explicit *metrical* structure, but it is not articulated rhythmically.

the Bach corpus demonstrates a case in which the perceived segment structure is conveyed almost entirely through pitch contour. We will discuss the problems this poses for MusiCog’s generation at greater length in Section 6.1.

Contemporary Music: Maxwell’s *Invidere*, for flute solo

As a final test, we trained MusiCog on a contemporary solo flute work, *Invidere*, by James B. Maxwell, and generated 10 works of approximately 112 measures duration. Complexity and expectancy for the original work, the MusiCog generations, and 10 random melodies (112 m. each), are plotted in Figure 5.34. Because the Maxwell “corpus” is represented by a single work, we did not perform significance testing. An excerpt of MusiCog’s 10th generation can be seen in Figure 5.35, for which we can make the same general observations. Formally, the generated melodies once again lack the symmetry and repetition structure of the source work (though symmetry in particular is a less important attribute in of the Maxwell than the previous training corpora), and the rhythm is again less regular—an excerpt of the Maxwell is shown in Figure 5.36. Harmonically, the quality of chromatic movement that typifies the implied harmonic structure of the source work has been replicated in the generated material. We also see frequent motive-level chromaticism, indicating that pitch content has not generally been quantized. This suggests that the PE’s mode induction must have frequently disabled pitch quantization, due to low confidence ratings, as discussed in Section 4.2.1. It is also worth pointing out that, with this music, the higher degree of chromaticism and lack of implied functional/tonal harmony are generally better suited to MusiCog’s interval-based generation than the other corpora.

Although the original work is rhythmically complex, its rhythmic figures do generally align with simple divisions of the beat (e.g., downbeat, eighth, sixteenth) or along clear n -tuplet subdivisions (e.g., aligned with the 2nd or 3rd triplet in a beat). In Figure 5.35, however, we see that rhythms are often subdivided in complex ways, with figures beginning at counter-intuitive metric/beat positions (e.g., on the 2nd thirtysecondth-note). Of course, this is partly a symptom of the rhythmic transcription process involved in converting the generated MIDI file to common music notation, but it is likely exacerbated by the rhythmic recombination process, which does allow for the generation of novel rhythmic values, not encountered in the corpus; a phenomenon we discuss in more detail in Section 6.3.1.

As with the previous corpora, another notable difference between the Maxwell work and the generated examples is that the latter do not demonstrate the clear high-level structure of the original. Specifically, whereas *Invidere* has several distinct sections of contrasting

material, the generated examples tend to spend more time exploring one or two types of material. This is likely due to the inability of MusiCog to extract high-level structure beyond the phrase level, due to limitations of WM capacity. It is worth note, however, that these limitations are not particularly inconsistent with those of human subjects. In more complex music, involving large-scale forms, it is part of the function of music notation and the musical score to provide a kind of surrogate memory, allowing composers to reflect upon previous material in a manner that would be quite impossible within the confines of working memory alone. The score allows composers to discover connections between motivic materials, and build complementary and/or contrasting content, in a manner that would be impossible for any musician with average skills of memorization and recollection.

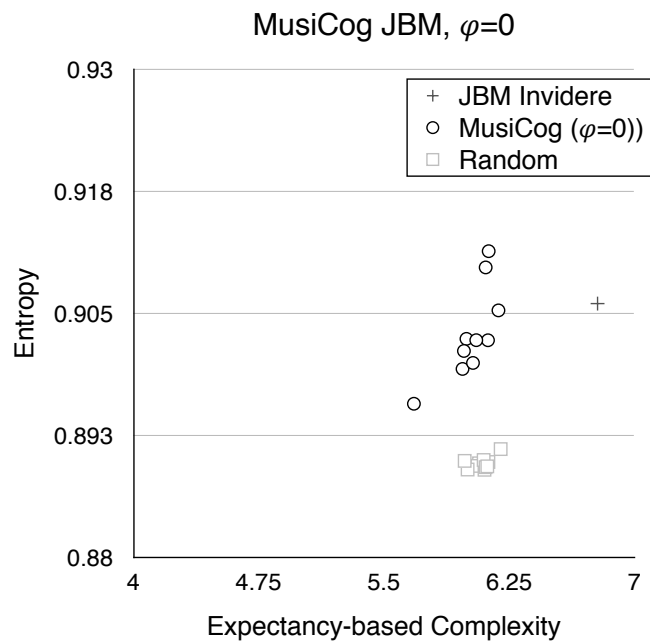


Figure 5.34: Plot of the complexity and entropy ratings for Maxwell's *Invidere*, MusiCog's generations, and 10 random melodies.

The image shows five staves of musical notation for a flute solo. The music is written in treble clef with a common time signature (C). The first staff starts with a measure of rest followed by a series of eighth and sixteenth notes, including triplets and sixteenth-note runs. Fingerings are indicated by numbers 1-5 above the notes. The second staff continues with similar rhythmic complexity, featuring sixteenth-note patterns and slurs. The third staff shows a transition to a more melodic line with slurs and fingerings. The fourth staff features a series of eighth notes with a slur and a fingering of 5. The fifth staff concludes with a series of eighth notes, including a triplet and a fingering of 3.

Figure 5.35: Score excerpt from MusiCog's 10th generation when trained on Maxwell's *Invidere*, for flute solo, showing exaggerated rhythmic complexity.

The image shows four staves of musical notation for a flute solo, labeled 'Flute', 'Fl.', 'Fl.', and 'Fl.'. The music is in 4/4 time. The first staff includes dynamic markings *f*, *p*, *ff*, and *fff*, along with a note marked with an asterisk and the instruction '* lip gliss. ad lib'. The second staff has dynamic markings *f*, *p*, *ff*, and *p*. The third staff has no dynamic markings. The fourth staff has dynamic markings *ff*, *f*, *ff*, *ppp*, and *pp*. The notation includes various rhythmic patterns, slurs, and articulation marks.

Figure 5.36: Score excerpt from Maxwell's *Invidere*, for flute solo.

Chapter 6

Discussion: Autonomous Composition in MusiCog

6.1 Discussion of Test Results

While statistically significant differences were found between MusiCog's style imitations and the corpora in all cases, score analysis revealed important similarities of both style and form. Consistent with our notion of MusiCog as a “musically naïve” composer, the system had increasing difficulty replicating styles that drew on increasing levels of music theoretical knowledge. This can be seen in Figure 6.1, which plots the complexity and entropy values for all corpora and all MusiCog generations. Here we clearly see increases in both complexity and entropy as the music becomes more theoretically derived; specifically, as the music becomes more dependent upon an underlying harmonic framework. Also of note is the fact that, as the complexity of the encoded music theoretical principles increases (e.g., dependence upon explicit or implicit harmonic structure as in the Jazz and Bach corpora, or deliberate subversion of strong cues for pulse induction or tonality/scale induction, as in the Maxwell), the overall difference between MusiCog's generations and the corpora becomes more pronounced. For example, when viewed in relation to all the corpora, the Folk groups (i.e., corpus and MusiCog) together form a well defined cluster, differing only in their internal distributions within that space. Likewise, the Jazz corpus shows a clear area of overlap with MusiCog's output, indicating that several melodies within the MusiCog group have replicated the style with relative success. However, for both the Bach and the Maxwell works, the MusiCog groups clearly occupy their own independent clusters.

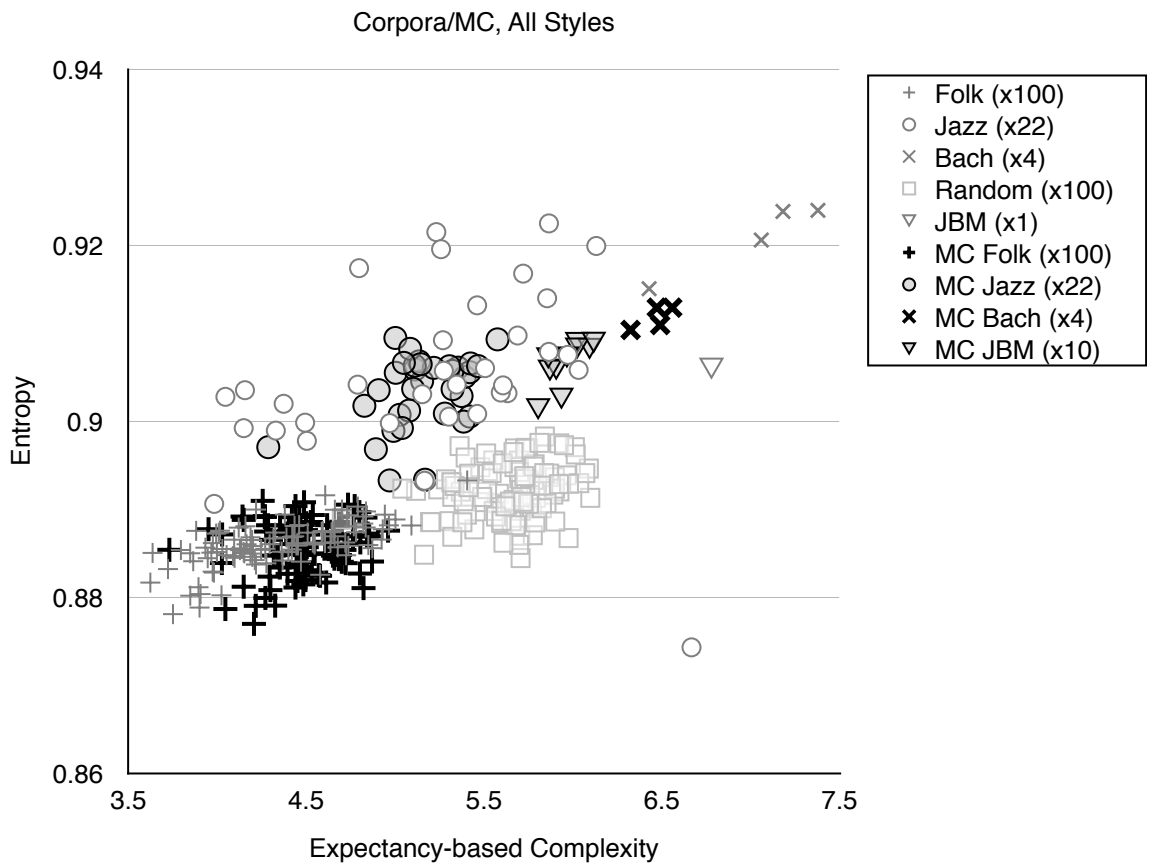


Figure 6.1: Complexity and entropy for all corpora and all MusiCog generations.

Looking at the overall plot, however, it is worth noting that all MusiCog groups form well defined clusters of “styles,” suggesting that MusiCog is able to capture the internal *differences* between corpora, and is replicating these differences during generation. Also of note is the fact that both the original corpora and the MusiCog groups are distributed across the complexity/entropy space in a similar manner, with the Folk corpus/MusiCog pair at the lower left, and each pairing of more complex styles moving toward the upper right¹. Once again, as the styles become formally more complex, MusiCog’s capacity for style imitation decreases. Interpreting this result from the perspective of the listener, it could be said that the more musically developed corpora are not as immediately accessible to perception as the simpler corpora, and require either an accompanying harmonic/chord progression, or explicit musical knowledge and/or experience for comprehension. MusiCog’s Bach generations indicate further problems, which we will discuss in detail now.

Problems encountered with the Bach corpus

In Section 5.4.2 we noted that MusiCog’s Bach generations demonstrated rather serious violations of style, primarily through the introduction of exaggerated syncopation. We suggested that this was due to the lack of hierarchical rhythmic structure in the Bach sample works, which contained extended passages of isochronous rhythmic values. In such a structure, segment boundaries are conveyed only through pitch contour, and not by the combination of pitch and rhythm, as in the other corpora. If we consider such a compositional structure from the perspective of recombination, we see that the stylistic requirement of maintaining continuous isochronous rhythms is actually quite a narrow constraint. This can be seen in Figure 6.2, where an isochronous sixteenth-note rhythm is presented, with an arbitrary segment structure added manually—i.e., the kind of segmentation that might be created by imposing a pitch contour on the isochronous rhythm. Below the segmented isochronous sequence are three arbitrary recombinations of the four segments, **A**, **B**, **C**, and **D**. In each case, the boundary position of the segment within the beat is maintained. We see clearly that none of the recombined sequences of segments can satisfy the style constraint of maintaining the isochronous rhythm. Extending this principle to the recombination of pitched material, Figure 6.3 shows a similar outcome, demonstrated according to MusiCog’s generative process of recombining boundaries and terminals. It will

¹The Jazz corpus poses somewhat of an exception, by way of its generally high variance, likely caused by the inclusion of subgenres (e.g., “Big Band,” “BeBop”) within the corpus.

be noted that the resulting syncopation is similar to that found in MusiCog's generations when trained on the Bach corpus (see Figure 5.33, Section 5.4.2).

Figure 6.2: Arbitrary rhythmic recombination of isochronous rhythmic segments produces a syncopated overall phrase structure.

It could be argued that there is a kind of implicit syncopation of phrasing in Bach's music, which is deliberately obscured by the isochronous sixteenth-note rhythms. Indeed, looking at the Bach excerpt in Figure 6.4, we see that the pitch contour does, in fact, generate a kind of "virtual syncopation," reminiscent of the "virtual polyphony" outlined in Section 3.2, whereby large interval leaps and changes of pitch contour jump out against the isochronous context. These events create "contour accents" [146], driving our attention away from the underlying pulse in a manner that mimics the effect of conventional rhythmic syncopation. This kind of writing is by no means common in the history of music. Rather, the common practice—particularly in monody, melody, or other monophonic forms—is to articulate phrasing through variation of *both* pitch and rhythm. Viewed in this light, it could be argued that the choice to create music that denies the role of rhythm in melodic segmentation, yet still creates the kind of rhythmic vitality syncopation provides, represents a deliberate compositional strategy. Though a somewhat hapless result when viewed from the perspective of musical style imitation, MusiCog's tendency to make Bach's underlying syncopation explicit is a curious byproduct of its recombinant compositional process.

Before we conclude our discussion of MusiCog's difficulty replicating Bach's writing, it is worth pointing out that, since rhythm has a strong influence on segmentation, MusiCog's

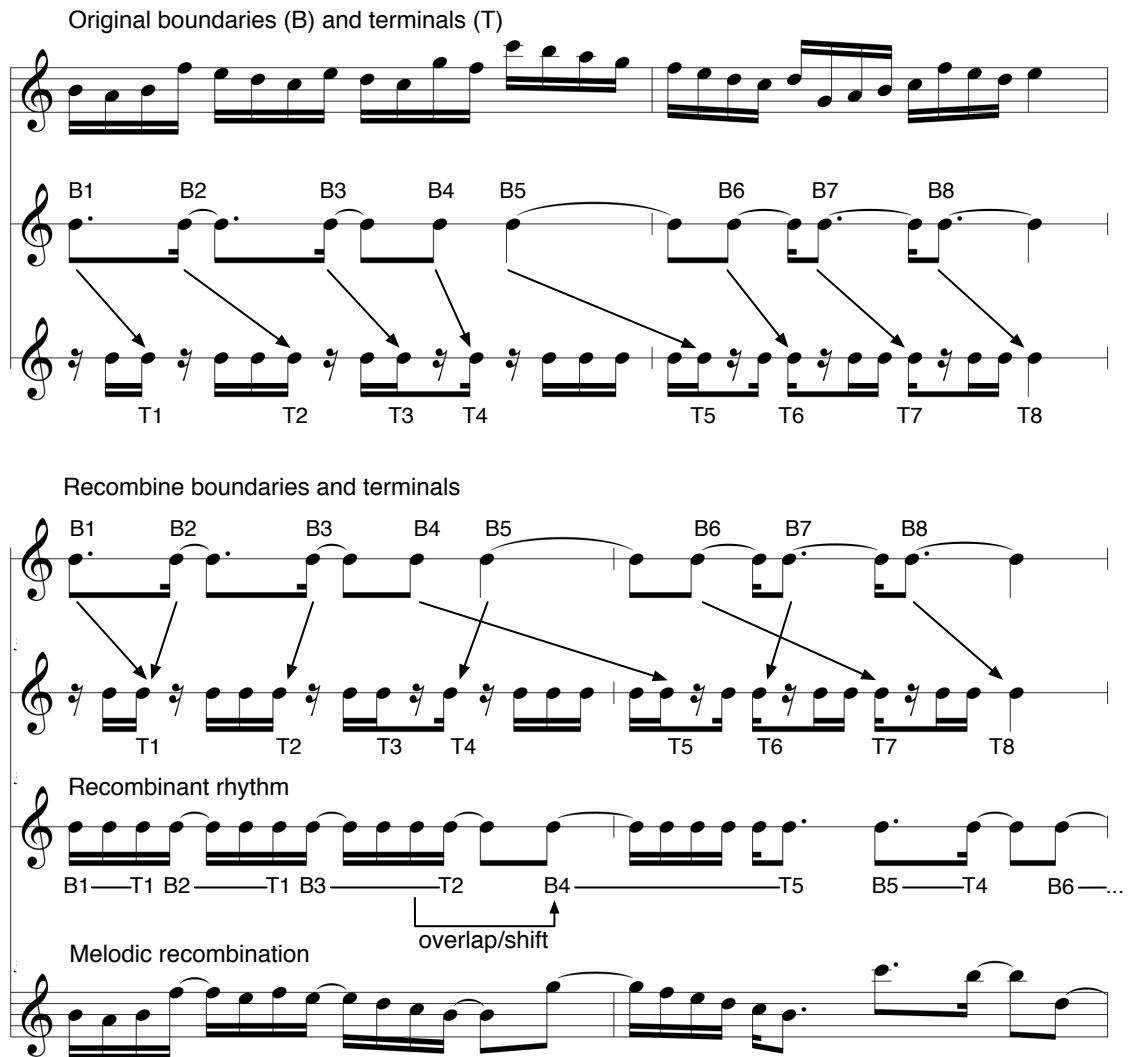


Figure 6.3: Result of segment recombination from an isochronous musical source.

arbitrary fragmenting of rhythmic structure would also introduce discontinuities between the hierarchical structure “intended” by the PM, and that “perceived” by the PE (as discussed in Section 4.2.4). Such discontinuities cause the inferred states in LTM to differ from the states used to generate the preceding segments in the PM, leading to complex and somewhat chaotic behaviour, as MusiCog jumps between different “interpretations” of the same material. This kind of behaviour would likely further impede MusiCog’s ability to replicate the corpus.

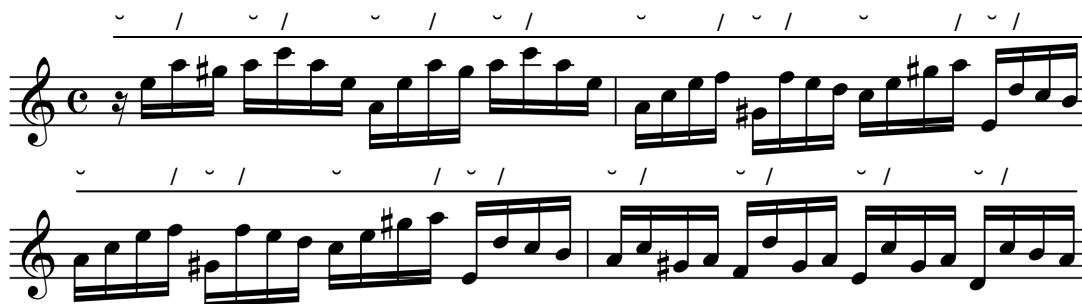


Figure 6.4: “Virtual syncopation” in the Allemande from Bach’s A Minor Partita.

6.2 Strengths of an Integrated Approach

The integrated approach to modelling music cognition taken in MusiCog has a number of benefits, which we observed both during development and in subsequent testing. Perhaps the most obvious is the fact that the modular architecture has allowed us to design and test specific perceptual/cognitive functions in a given module, without directly altering other modules. Although the various modules exert direct, and in some cases quite profound influences upon one another, they are formally decoupled, so that their functions can be examined independently. Further, the compartmentalization of functions provides a clear conceptual framework for considering the relationship between ideas from the music psychology literature and their implementation in the model (e.g., melodic segmentation in the PE, chunking in the WM, etc.).

It has also been valuable to discover, through the interaction of modules, areas where underlying assumptions of our earlier approaches were conceptually misleading. For example, when the CM structure was originally designed [162] our intention was to build a hierarchical model with the capacity to learn complete musical works. However, with the introduction of a distinct WM module, designed to approximate the short-term music retention capacities of listeners, a more concrete limitation was placed on sequence learning. It soon became clear that an LTM representation that learned only from the sequential presentation of musical events, and was thus restricted to associative chaining and other forms of serial order memory [102, 133], should be limited in the amount of hierarchical structure it can learn [216]. In practice, the span of contiguous, hierarchical musical structure learned by MusiCog is roughly on the order of an average melody, or 12 to 24 events, depending on the specific segment structure of the music. As such, MusiCog is unable to learn complete works as hierarchical sequences; a similar situation to human listeners, who rely on structural factors [169], various forms of memory (i.e., beyond cued recall),

and alternate learning strategies to memorize complete musical works² [35, 90]. Similarly, the importance of working memory in the compositional process was not obvious to us until we switched to a modular approach. Though previously investigated with regard to literary composition [166, 167], the manner in which music composition enlists the working memory faculties of composers has received virtually no attention. By creating a feedback system in which working memory is directly implicated in compositional decision making, we were able to produce a simple form of cognitively-grounded motivic exploitation, which is relatively rare in corpus-based, real-time, autonomous generative music systems—the Anticipatory Model of Cont et al., (Section 3.6), Smith and Garnett’s hierarchical neural network model [215], and Collins’ Racchman-Oct2010 and Racchmaninof-Oct2010 systems [42] are of interest in this regard—at least without the specification of high-level music theoretical heuristics.

Another important aspect of the integrated approach concerns the manner in which information is structured prior to storage in LTM. For example, by using the PE to model implicit knowledge of melodic structure (i.e., via gestalt-based melodic segmentation), we were able to limit the number of distinct patterns stored in LTM. This is in contrast to other models of musical knowledge, like n^{th} -order Markov models, suffix trees, and certain n -gram models [73], which store all unique substrings of a given input string. Having clear segmentation points and a limited vocabulary of well-formed segments greatly facilitated MusiCog’s acquisition of hierarchical structure. Additionally, the limited capacity of the WM provided a natural sliding window for the incremental construction of such hierarchies, and the chunking model served to further constrain the types of hierarchical forms that would ultimately be stored in LTM.

Finally, having a dedicated executive module for composition, modelled by the PM, has allowed us to involve both working memory and long-term memory—and the interaction between the two—in the generation of new musical materials. Because the contents of WM are regulated by the PE’s processing of previous output via feedback (a form of self-regulation), the structure of WM is highly dynamic in nature, allowing MusiCog to create novel possibilities from the corpus-based knowledge stored in LTM. Also, since pitch quantization is performed in the PE, as a final step prior to output, material processed through feedback can be further distanced from the WM/LTM representations from which it is derived, via pitch/interval transformation. This again offers possibilities for the generation of

²Cases do exist in which listeners have been able to recall complete compositions from memory, but such cases are rare and generally attributed to savants, or exceptionally gifted individuals [179].

novelty and, in cases where feedback learning is enabled during generation, allows MusiCog to further expand the space of possibilities represented by the training corpus. The potential of this kind of feedback and self-regulation has only been touched upon in the current implementation.

6.3 Questions and Challenges

Broadly speaking, stochastic generation from MusiCog revealed problems related to a lack of repetition structure and formal symmetry, exaggerated rhythmic syncopation, and in the case of the Bach corpus in particular, stylistically inconsistent implied harmonic structure. On the other hand, certain important properties of human composed music were observed; in particular, the variation and development of generated motivic materials. Also, as observed in a recent comparative study by Gonzalez Thomas et al. [92], MusiCog was able to better approximate the statistical regularities of training corpora than basic implementations of a Variable-Order Markov Model and a Factor Oracle [9]. This relative success is likely due to the CM's ability to encode temporal dependencies between both adjacent and non-adjacent events in the training works, as a result of its cognitively-grounded exploitation of melodic segment structure during learning and generation. MusiCog was also shown to generate tonal implications, when trained on tonal material, with a regularity that could not be attributed to its pitch quantization function alone (see Section 5.4.1). However, in most cases the musical weaknesses of MusiCog's compositions became obvious upon inspection of the generated musical scores³. Although the shortcomings of the generative approach used are largely attributable to an over-emphasis on stochastic decision processes and the lack of explicit music theoretical and compositional knowledge, certain problem areas can be more directly attributed to MusiCog's generation algorithms. We will discuss these issues, and some possible solutions, in greater detail now.

6.3.1 *Stylistically inconsistent rhythmic complexity and syncopation.*

Due to the recombinant nature of MusiCog's generation processes, the system is capable of creating transitions not contained in the training corpus. A simple example showing the generation of novel pitch intervals is given in Figure 6.5. Here we see that, given the boundary relationships contained in phrases 1 and 2, by swapping the terminals of the first

³It should be noted, of course, that listening would also reveal these weaknesses.

segments, so that **a** is replaced with **c** and **c** is replaced with **a**, we can produce both a repetition (i.e., interval 0) and an ascending major 2nd (“+M2”), neither of which are present in the example phrases. Although this is desirable as a potential source of novelty, allowing MusiCog to expand its representational space, it is also a potential source of style violations during generation. In the case of pitch intervals, serious problems are not likely to arise, since any training corpus of reasonable size will likely contain a complete set of useful musical intervals. However, in the case of rhythm, novel intervals (i.e., IOIs) arising through recombination of this kind can pose more serious problems.

Figure 6.5: A simple example of MusiCog generating novel intervals through recombination. By swapping the terminals of the first segments, so that **a** is replaced with **c** and **c** is replaced with **a**, we can produce a repetition (interval 0) and an ascending major 2nd (“+M2”), neither of which are present in the example phrases.

In general, novel transitions can occur in MusiCog when two (or more) phrases share a common L_2 boundary transition (i.e., in the CM) that joins together different L_1 segments. In such cases, the L_2 transition becomes a kind of ‘pivot’⁴ between training phrases, allowing the L_1 segments to become interchangeable during generation. A simple example for rhythm is shown in Figure 6.6. Here transitions (A, B) and (C, D) express the same L_2 boundary relationship (-1), but use L_1 segments with different beat divisions, and different boundary positions within the beat. During generation, one boundary may select an L_1 sequence of three eighth-note triplets, while the following boundary selects an L_1 sequence of

⁴The analogy here is to the function of “pivot chords” in harmonic modulation. Pivot chords are harmonic structures shared between two keys, but having different harmonic functions in each key. The dual function of the pivot chord is used to generate harmonic ambiguity, easing the authority of the previous tonality, and supporting the listener’s expectation for harmonic resolution in the new tonality.

four sixteenth-notes. If both boundaries fall on even beats then the change in beat division will align with the beat structure of the music, and a fairly common form of rhythmic variation will be produced, as in 6.6a. However, if the boundary of the second segment falls on a weak, up-beat position—for example, on the last sixteenth-note of the beat—the resulting pattern will modulate the beat division independently of the beat, leading to a complex, and unexpected rhythmic pattern, as in 6.6b. This kind of recombinant process was likely involved in the problems observed in Section 5.4.2, where MusiCog’s generations demonstrated stylistic inconsistencies due to exaggerated rhythmic complexity, as was particularly apparent when trained on the Maxwell work.

Figure 6.6 illustrates rhythmic transitions through recombination. The top staff shows four segments (A, B, C, D) with a 3-note triplet and a -1 interval. Below are three examples:

- a) A+B with a 1/8th triplet IOI
- b) A+D with a novel IOI and "Beat division modulates within the beat"
- c) A+D: alter D with IOI 0 and "Fails to retain the boundary relationship"
- d) A+D: alter rhythm with IOI -1 and "Loses the rhythmic character of A"

Figure 6.6: It is possible for MusiCog to create novel (and in some cases stylistically inconsistent) rhythmic transitions when segments with different beat divisions are recombined during generation.

The process of resolving such rhythmic conflicts, though simple and intuitive for human composers, is by no means straightforward for a computational system. Examples 6.6c and 6.6d show two human-composed solutions to the recombination of patterns **A** and **D**, both of which require some degree of alteration of the patterns themselves. In 6.6c, pattern **D** is altered by removing the first note. Despite proposing a fairly dramatic alteration of the pattern, this is a reasonable solution because it maintains the pitch accent structure of the original theme by ensuring that the C5 remains on the beat (for this reason, shifting the entire motive forward, so that the initial B4 lands on the beat, would be a less satisfactory solution). It’s worth note, however, that removing the initial B also alters the boundary

relationship, resulting in a flattened L_2 pitch contour. The solution shown in 6.6d takes a different approach by altering the beat division of **A**, so that it matches the sixteenth-note division of **D**. This solution maintains the pitch structure and length of both **A** and **D**, while also retaining the boundary relationship. However, it has the negative effect of reducing the rhythmic complexity of the passage, sacrificing the suspended quality of the triplet pattern in **A**, and losing the feeling of acceleration provided by the shift to sixteenths in **D**. However, both solutions are arguably better than the purely recombinant solution produced by MusiCog. It would be possible, of course, to include a function to test for these sorts of collisions between different within-beat rhythmic divisions in the stochastic selection process, removing options that would result in such mixed patterns before a selection was made. Assuming a reasonably well-trained CM, such a solution could provide an effective method for avoiding such rhythmic problems.

With regard to the problem of exaggerated syncopation, we observed this when trained on the rhythmically complex material in the Maxwell work, but also, most notably, when trained on the Bach corpus. In the former case, problems with novel transitions generated through rhythmic recombination—in particular, the recombination of regular and n -tuplet patterns in weak, up-beat positions—appeared to be the root cause. In the latter case of the Bach corpus, however, the stylistic requirement of producing extended passages of isochronous rhythms—and the extremely tight constraint this poses for a recombinant system—is certainly the root cause (as discussed in Section 6.1). The solution, however, is not particularly obvious. One option would be to remove the beat-based representation provided by the Beat ED, and use a simpler IOI representation. But this cannot be done without sacrificing the segment-level beat structure, which, in the vast majority of cases, would be too great a sacrifice (see Section 4.1). Another option would be to group rhythmic material by beat, as is done by Eigenfeldt [81], rather than by the perceptual segment boundaries. This approach, while advantageous for the rhythmic generation process, will generally fail to support the segment structure identified by the PE (which frequently crosses beats), and is thus inappropriate for a cognitive model like MusiCog. Yet another option would be to use low-order (e.g., zeroth- and first-order) Markov models for rhythmic generation, while retaining the segment-based approach used in the CM for generating pitch structure. However, although it is likely that such an approach would perform better with this particular material, it does not *guarantee* the isochronous structure required for style imitation of such tightly constrained (rhythmic) writing. Further, as with the option

of using simple IOI values, this option likewise risks sacrificing the segment structure afforded by the current system. However, it is worth pointing out that training on a larger corpus—even one that contained further examples of this kind of isochronous rhythmic writing—would greatly improve MusiCog’s chances of replicating the style. That is, it may simply be a matter of being presented with more combinations of segments that satisfy the constraint of maintaining isochronous rhythm, thereby increasing the probability of satisfying the constraint through purely stochastic processes.

The point we would like to stress is that, in this music, Bach has deliberately avoided the common practice of articulating melody through the joint variation of pitch and rhythm⁵. In this sense, the rhythmic structure represents a kind of conceptual “strategy”; i.e., it is a formal contrivance used to draw attention to a specific compositional objective. In order to produce a musical work like this, a generative system would likewise have to form a similar abstract compositional objective. This is clearly not something that a musically naïve system like MusiCog can do, and it is a much larger question how such a system should be designed. Of course, this corpus is of interest because it exacerbates a weakness in MusiCog’s approach that might not be particularly obvious with other models. However, it is also of interest because it points out that, in a certain sense, MusiCog is a fairly conventional composer. That is, its compositional understanding is so constrained by bottom-up perceptual factors, that it is fundamentally incapable of generating music that denies these factors (as is the case with Bach’s extended isochronous rhythms), even when presented with examples that establish the musical viability of the practice. Certainly we can well imagine that even the most naïve human composer, when asked to imitate this style, would quite naturally produce an isochronous rhythmic pattern. In this sense, MusiCog’s weakness extends beyond its limitations as a “musically naïve” composer. Rather, there is a level of generalization that is missing; an ability to build an overall conception of the compositional task, and to ensure that the most obvious elements of the musical style are emulated. The isochronous rhythmic structure is arguably the most obvious feature of this music.

⁵In designing a study to isolate the role of pitch in melodic accent structure, Huron acknowledges the difficulty of locating example works that demonstrate this kind of continuous isochronous writing. It is no coincidence that Bach featured prominently in Huron’s test corpus [107].

6.3.2 Inability to reliably produce tonal melodies, when trained on tonal materials.

Given that MusiCog has no explicit model of tonality, and includes no deliberative routine for establishing (or supporting) a tonal centre, the fact that it appears to be capable of generating tonal implications when trained on simple tonal source material (see Section 5.4.1) is a significant finding. Although the pitch quantization mechanism clearly plays an important role in producing this behaviour, the constraint of generating within a diatonic scale cannot itself explain MusiCog's capacity in this regard, as was shown in Section 5.4.1. Rather, such tonal structure must arise as an architectural phenomenon, resulting from 1) the segment structure established by the PE, 2) the chunking provided by WM, 3) the hierarchical structure learned by the CM/LTM (in particular the relationships between segment boundaries learned at L_2), 4) the PE's induction of mode/tonality in response to feedback from the PM, and 5) the PM's function of using the induced mode, once established, to quantize pitches⁶ for the remainder of the generation.

Of course, tonal composition is a complex problem, since it is clear that, while the impulse to identify a tonal centre is a fundamental behaviour of music cognition, it nevertheless should not be an *a priori* assumption of a general purpose style imitation system. Also, subtle chromatic alterations of key are common, particularly in music of the baroque, classical, and romantic periods, where they are often used to provide colour, or for tonicization of non-tonic chords (e.g., for supporting harmonic modulation)⁷. Indeed, it was this desire to maintain local chromaticism that prompted our choice of a segment-based pitch quantization approach, thus allowing generation to return to the previously induced scale when diatonic segments are detected. However, such an approach does prohibit the possibility of modulation—even for brief, intermediate (or “false”) modulations, as are often found in Bach's music, for example. One possible solution might involve generalizing the mode induction function to provide a confidence rating for the diatonicism of the passage, then providing independent confidence levels for all possible tonics. As in the current implementation, the “diatonicism” confidence rating could be used to enable/disable pitch quantization. The individual tonality confidence ratings could then be used to determine

⁶Recall that pitch quantization will not be applied if the generated segment elicits a low confidence rating from the PE's mode/tonality induction function.

⁷These traits became less common in 20th- and 21st-century music, when functional harmony fell out of favour, or in popular forms, where violations of key are rare. Jazz music has, for much of its history, utilized scale modifications as a means of providing character, often relating scale structures to the chords themselves, rather than to the tonic key, as is the practice in classical, folk, and popular forms.

the most probable tonal centre(s) at a given moment. In cases where multiple pitches indicated relatively equal confidence, any previously induced tonic could be deleted, thus modelling the perceptual ambiguity of “pivot” harmonies, and allowing the system to respond appropriately to modulatory passages. When confidence in a particular tonic was high, the system could rotate the induced mode to the scale position of the perceived tonic and perform pitch quantization.

Currently, MusiCog lacks any heuristics for modelling stylistically consistent pitch registers during generation⁸. This can occasionally lead to uncharacteristically large registral ranges, as seen in some of the generated examples, as a consequence of compounding same-direction pitch contours at different levels of form; e.g., joining descending L_1 pitch segments with descending L_2 boundary transitions. Implementing some method for acknowledging the total pitch range when selecting pitch contours between boundaries could help to reduce the likelihood of such discontinuities of register during generation.

6.3.3 Lack of formal repetition structure and symmetry.

This is perhaps the most difficult problem for generative music systems to solve; to create convincing musical forms, with adequate repetition, variation, and higher-level structure. The challenge is to allow for the generation of novel patterns, while at the same time allowing previously *generated* (i.e., not necessarily learned) patterns to be quoted verbatim; a common feature in many forms of music. In general, the former problem is solved by stochastic selection mechanisms, while the latter has perhaps only been adequately solved by Cope’s systems [52], which use high-level, grammar-based musical knowledge to replicate this kind of formal organization. As was observed in Section 5.4, MusiCog is able to at least partially solve this problem, by exploiting WM contents, provided by its feedback architecture, to support the kind of motivic development observed in human-composed music. However, because MusiCog still allows for stochastic variation when exploiting motivic materials (i.e., by generating L_1 segments from the terminal nodes held in WM, rather than simply quoting the segments directly), in some cases the imitation is too distorted (i.e., from pitch/rhythm transformation) to produce convincing repetition structure.

However, perhaps a more serious impediment to creating the kind of repetition structure and symmetry found in the training corpora is MusiCog’s current lack of a metric induction (and composition) mechanism. Metric structure helps ground the expectation patterns

⁸The PM does include a routine for constraining pitch to a reasonable maximum range (approx. B \flat 2 to F6), however, this routine did not appear to be used in the generated examples.

found in many styles of music, allowing listeners to predict high-level formal patterns given only short periods of musical context. Of course, there is something paradoxical about this situation, since it is the musical structure itself that creates the sense of regularity we describe as metre. Clearly, from the composer's perspective, the relationship between metric induction and composition is dialogical. However, metric induction/composition alone would not ensure the ability to generate convincing repetition structure and symmetry. As discussed in section 4.2.3, MusiCog does encode patterns of repetition and symmetry at higher levels of the CM, but this structure is often lost during the selection of sublevel terminals and/or the extraction of paths leading to those terminals. Because terminals are chosen stochastically, as are their paths, the repetition suggested by the higher-level form is put into jeopardy each time a new stochastic selection is made. By contrast, in human compositional processes, simple forms of melodic development—restating the theme at the dominant level, for example—are not fundamentally stochastic, but rather appear to be rule-based, deterministic processes. Such processes produce variety by applying some fundamental transformation (in this case, transposition), but maintain unity by ensuring that this transformation is applied uniformly to the entire musical idea (i.e., segment or phrase). However, operations like this require that the musical idea itself remain invariant on some level, so that it can be recognized after the transformation has been applied. This suggests a fundamentally iterative process; i.e., the idea is first established, then subsequently altered via transformation. Such a process may be purely cognitive, without the need for explicit “stages” of compositional development (see Section 2.3), but it also cannot reasonably be explained by local stochastic selection.

It is also worth noting that MusiCog cannot currently learn formal structures that extend beyond the capacity of WM, and is therefore limited in the amount of high-level form it can represent. This limitation is not unfounded from a cognitive perspective, where it is clear that the kind of episodic, chaining memory currently modelled by MusiCog does not generally extend to higher formal levels. Indeed, methods for retaining such high-level form—and the specific structures used to represent high-level musical knowledge—are not currently well understood by the music psychology community. Most research in the area speaks of “high-level” form when dealing with musical passages only a few bars in duration, where the retention of non-adjacent temporal relationships implies a form of hierarchy, even if the temporal durations are not particularly long. But representation and retention on the order of complete musical works remains relatively unexplored.

The work of Deliège on cue abstraction and schematization (the formation of abstract representations of high-level form during listening) [57, 58], later investigated experimentally by Koniari [121], is directly related in this regard, showing that high-level cues are often abstract, invariant representations of general auditory patterns; “frequent trills,” “isolated sounds bound by groups of rapidly flowing sounds,” and so on [57]. However, Musi-Cog does not currently possess a mechanism for building the sort of linear memory for sequences of cues⁹ proposed by Deliège. In a related area, studies of the learning and memorization strategies of professional musicians [35] suggest that certain methodological approaches do exist for retaining and recalling attributes of high-level structure. In part these involve the recognition of important structural locations or events in a given work (e.g., via points of multiple closure [174, 216]), which can be used to cue memories of more local details. Additional factors include the use of motor, visual, historical, and “structural” memories [35]—a musical version of Rubin’s “narrative” memory: memory for the sequential organization and goal structure of a story [201]—to assist in building reliable mental representations of complete works. However, it is not clear whether such processes are also available to untrained music listeners, and if so, to what degree. Composers, of course, are only required to retain such high-level form in an abstract sense, having recourse to music notation and other technologies for capturing structures that extend beyond their capacity for memorization. Further, knowledge of musical form passed down through musical training also informs the compositional decision process in this regard.

At any rate, this problem of balancing local variety with formal unity (repetition structure and symmetry) puts into question theories of composition that propose a strictly predictive, probabilistic process. It seems reasonable to suggest that both stochastic/probabilistic and deterministic/rule-based approaches must coexist in human compositional thought, in order for such symmetrical structures to have become so prominent in music. Following Collins’ work [41], we see it as a fair assumption that the compositional process proceeds through iterative, hierarchically-related stages, in which initial decisions regarding high-level formal structure and local motivic pattern selection can be made stochastically, but will be subsequently fixed in place as concrete compositional *rules* to be followed during later stages of the compositional process. Indeed, all of the examples from the folk corpus used during the testing carried out in Section 5.4 demonstrate a similar degree of symmetry and repetition, and also a similar binary structure, suggesting that such formal principles are established

⁹The term “cue” is used here in Deliège’s specific sense, analogous to her notion of the “imprint”; a schematic, invariant representation of an abstract, perceptually categorizable musical structure.

with high probability at an early stage of composition, and are likely even considered a *priori* preconditions of the style itself. If this is the case, then such formal strategies are not “decisions” at all, in the conventional sense, but should rather be considered as concrete rules. Seen from this perspective, the process of folk song composition is more concerned with generating appealing and well-formed motivic segments, that fulfil the requirements of an inherited formal structure, than with any notion of “through composition,” in the contemporary sense. The same could be said for many of the conventional Western forms (binary, ternary, rondo, sonata), and also the majority of contemporary Pop and Jazz songs. Other forms, like the style of fantasia that has become commonplace in late 20th- and 21st-century composition, or forms of improvisation, also benefit from a degree of formal regularity, though cyclic structure and verbatim quotation are often avoided.

6.4 Creativity in MusiCog

Generation in MusiCog has thus far focused on a somewhat automatic, associative process, biased toward exploiting musical parallelisms. This approach, while useful for observing the behaviour of the model on an architectural level, does not yet explore the model's potential for creative output. Certainly, its capacity for generating novel, and in some cases quite acceptable, melodies suggests that the model demonstrates a degree of E-Creativity (see Section 2.5.1)—i.e., the ability to explore the musical space represented by a given training corpus. However, as was seen in Section 5.4, MusiCog's generated melodies demonstrated consistently lower variance than the training corpora, when measured in terms of complexity, entropy, and SIMILE's cognitively-grounded `opti3` measure, suggesting that it currently does not explore the musical space as thoroughly as it should. Although a single reason for this cannot be assumed, it is likely that the bias toward high-probability transitions during stochastic generation is a significant factor. Also, because it strives to support cognition through the exploitation of parallelism, but does not retain a memory for its previous output when starting a new melody, it tends to constrain the space of its musical development within generations, and cannot deliberately expand the musical space across generations.

Since the model will increase its CM structure when learning from feedback output (see Section 5.4.1, p. 147), it is reasonable to suggest that the model demonstrates basic P-Creativity; i.e., the ability to create output that is novel to the system itself. Simply put, if the generated material did not produce novel structures then it would not promote learning

in the CM upon feedback. Although it is perhaps a contentious claim, since MusiCog does have the ability to create novel transitions, and can subsequently learn from those novel transitions during feedback learning, it could also be said to at least possess the *potential* for T-Creativity; i.e., the ability to transform the musical space represented by a given corpus. This was observed to some degree in the feedback learning test (Section 5.4.1, p. 147), where MusiCog’s output was seen to produce a distinct, higher complexity space than the training corpus. However, it is important to point out that, while the system’s capacity for originality appears to be relatively high, its capacity for achieving quality remains relatively low, when evaluated against its training corpora. Similarly, its capacity for typicality (see Section 2.5.3) is also relatively low, due primarily to its inability to replicate the higher-level repetition structure and symmetry that characterized much of the training material.

Nevertheless, as a test of the “musically naïve” approach to composition proposed in the Introduction (Section 1.1), MusiCog performed acceptably, and demonstrated certain important traits of human creativity; in particular, the capacity to develop musical materials of its own devising (i.e., motivic exploitation). This behaviour can be attributed in part to the learned structure of the CM, but is predominantly due to architectural factors associated with the integrated approach. Because feedback between the PM and PE allows MusiCog to populate its own working memory with motivic materials of its own devising, enabling it to selectively re-use those materials during generation, the form of motivic exploitation demonstrated can be said to be modelled at an architectural level—a general goal in ICA design [130].

Chapter 7

ManuScore: Cognitively-Grounded Computer-Assisted Composition

In contrast to the totally autonomous composition demonstrated by MusiCog in the previous chapter, Computer-Assisted Composition (CAC) focuses on the use of computers as compositional aides for human composers. Taken in its complete breadth, the field ranges from the most esoteric research projects to the most highly profitable commercial music production platforms. The goals of different CAC packages can therefore be quite distinct, in some cases serving to streamline a set of familiar production techniques (recording, editing, scoring, and so on), and in others proposing to unlock the creative potential of composers, helping them move beyond the constraints of their musical habits. In developing his notion of composition theory Laske clearly placed his focus in the latter category, seeking to better understand compositional processes, so that composers could break with the past and open up new territories of musical possibility. As discussed in Section 2.1, Laske's theory focused on three fundamental principles: 1) *competence*: knowledge of the materials and syntax of music, required for the conception of musical ideas; 2) *performance*: the practical application of accumulated musical knowledge (competence) to create musical forms; and 3) the *task environment*: the field of action in which performance draws on competence for the invention of musical works. In the context of CAC, the task environment is embodied by a computer and its hardware/software. Laske felt that, due to the inherent pliability of computer software, the computer-based task environment enabled composers to access the "virtual music" of their imaginations in a manner unbounded by musical traditions. He identified the process of conception, design, implementation, and production

of the task environment, and its iterative development throughout the compositional process, as the “compositional life cycle” [137]. It was Laske’s feeling that software developed through a compositional life cycle could gradually begin to embody the musical knowledge of the composer in an explicit, analyzable, and extensible way.

Early CAC tools like Koenig’s PROJECT systems [134], or Truax’s POD systems [230], took a top-down approach to CAC, in which high-level concepts were expressed parametrically, or graphically, and the software was responsible for generating numerical representations (or electronically synthesized performances) of musical output. Two important observations can be made about such systems: 1) They are not corpus-based, and thus will not generally maintain an explicit connection with the user’s musical past, and 2) They deal with musical concepts at a high level of abstraction, and introduce significant non-linearities into the compositional process (i.e., by expressing musical ideas through numerical representations, requiring subsequent interpretation by the composer). In this manner, they distinctly separate composition from the act of listening; an idea that Laske very much supported (see Section 2.1), believing that such a division could lead to unbounded musical invention.

Although high-level generative functions are beginning to appear in commercial CAC packages¹, the majority of such programs—Digital-Audio Workstations (DAWs) and MIDI sequencers—are essentially bottom-up systems, which fulfill the basic tasks of recording musical performances for subsequent editing and manipulation, or transcribing and electronically “performing” musical scores. Although applications of this type are often equipped with extensive feature-sets, directed toward simplifying and streamlining the workflow for such tasks, their fundamental purpose is to *record*². Making up the middle-ground, there are an increasing variety of CAC tools that demonstrate different (and often quite novel) forms of musical representation, and introduce varying degrees of interactivity into the compositional process ([165, 234, 253] etc.). Among this class one could also include the increasing variety of music programming languages, graphical or otherwise ([10, 224, 238], etc.), which offer a potentially infinite variety of CAC *tools-to-be*, and propose potentially infinite mixtures of top-down/bottom-up control.

¹The “drummer” in Apple’s Logic Pro X software provides a highly sophisticated example of generative algorithms in a commercial context.

²For this reason some may not consider these to be genuine CAC tools. However, even the rather basic advent of non-linear editing had a significant influence on popular music production, suggesting that it is reasonable to include these basic packages under the CAC heading.

For the bottom-up tools, competence and performance are essentially unchanged from the traditional requirements of music-theoretical knowledge, instrumental performance ability, skill in instrumental music transcription, and so on. With the top-down tools, the demand placed on competence is shifted (and potentially increased) by the emphasis on abstraction, while performance becomes focused on the interpretation of numerical representations of musical materials [136], and/or on the comprehension of metaphorized descriptions of musical concepts: “density”, “timbral trajectory”, and so on [230].

7.1 Design Motivations Behind ManuScore

Our initial goal in designing ManuScore was to create a music notation-based CAC tool for music-literate composers, who might already possess a developed musical language and bottom-up compositional practice, but were interested in exploring a top-down interaction with their musical ideas. In contrast to Laske’s goal of freeing composers from musical tradition, ManuScore was built to acknowledge the user’s existing musical practice, so that working with it need not impose any dramatic change in a composer’s musical language or compositional output. The intention was to create a task environment to augment a composer’s practice, not necessarily to dramatically alter it, and certainly not to completely automate it. In this sense, the system could be aligned with Cope’s CUE software [50], which draws from a music recombinance database to offer “continuations” and developments of musical ideas introduced by the user.

Adding to this general conception of a non-interfering, interactive CAC tool with corpus-based generation, ManuScore was also an investigation into the notion of “object-oriented” composition [204]. Our conception of object-orientation focuses on the notational aspects of musical ideas; i.e., on what can be captured in a musical score, what the various notational structures represent to the composer (i.e., what is their musical “objecthood”), and how these structures might “inherit” from one another in the developing composition. In this sense, an *object* in ManuScore is essentially a “gestalt”; an identifiable, holistic item, or concept. To whatever degree possible, ManuScore was designed to help composers explore musical ideas as gestalts, and to represent them accordingly in their compositional task environment. This approach connects ManuScore to programs like PatchWork (or PWGL [139]) and OpenMusic [10], which also help composers interact directly with musical concepts, though its focus on notational elements (i.e., leaving aside numerical operations) in the user interface clearly sets ManuScore apart.

Because it is intended as a CAC tool, generation in ManuScore focuses on the notion of “continuation”—i.e., extending musical fragments introduced by the user—and uses Musi-Cog as a built-in intelligent musical agent for providing monophonic melodic continuations. However, our long-term goal with ManuScore is to implement real-time, interactive generation, so that musical ideas may also be explored through listening and improvisation, not just through the manipulation of scored musical objects.

7.2 ManuScore Design & Features

The Graphical User Interface (GUI) for ManuScore is designed to emulate a “pencil and paper” workflow, while attempting to maintain a balance between power and flexibility. Whenever possible, it utilizes the standard ARROW UP/DOWN/LEFT and RIGHT keys for moving objects, selecting accidentals, toggling articulation markings, and so on, requiring the user to remember only a limited set of possible interactions when learning the software.

7.2.1 An Open Musical Space

At launch, the ManuScore GUI presents the user with an empty space—a blank canvas, so to speak. The background displays faint vertical guides, which act as a temporal grid for entering musical events. The grid does not strictly follow the conventions of musical time signatures. Rather, it presents a *visual* guide to subdivide the musical space, serving a similar function to the grid systems found in graphics and drawing software packages, and provides “snapping” functionality to assist the user in entering rhythmically precise material. Objects can be moved independently of the grid, so that events can be placed at any location in musical time.

At the top of the score window, “Metric Markers” can be inserted, allowing the temporal grid to be subdivided in arbitrary ways. It is worth noting that this division is strictly graphical, and imposes no formal restrictions on the music itself. The numbers in each marker indicate a grouping/subdivision of time. The top two numbers are conceptually analogous to a conventional time signature, while the bottom number indicates the number of “beat divisions” used for object snapping, and is thus capable of producing any n -tuple subdivision. Figure 7.1 shows a sample score with two Metric Markers added. It will be noted that the markers only alter the grid for the rhythmic space *following* the marker’s position. This can be seen in Figure 7.1, where what *appears* to be a $\frac{4}{4}$ time signature is cut short

by a $\frac{7}{8}$ signature, inserted part-way through the first measure. In conventional notation software, replicating the musical meaning of this structure would require the user to completely redefine the metrical notation of the music (and in most cases, to delete and re-enter the musical passage). However, because the temporal grid in ManuScore is essentially independent from the contents of the staves, this sort of structure can be created at any time, without altering the existing musical material. In this sense, ManuScore’s rhythmic representation offers a smooth temporal space for composition, as opposed to the highly structured metrical space of conventional notation software.

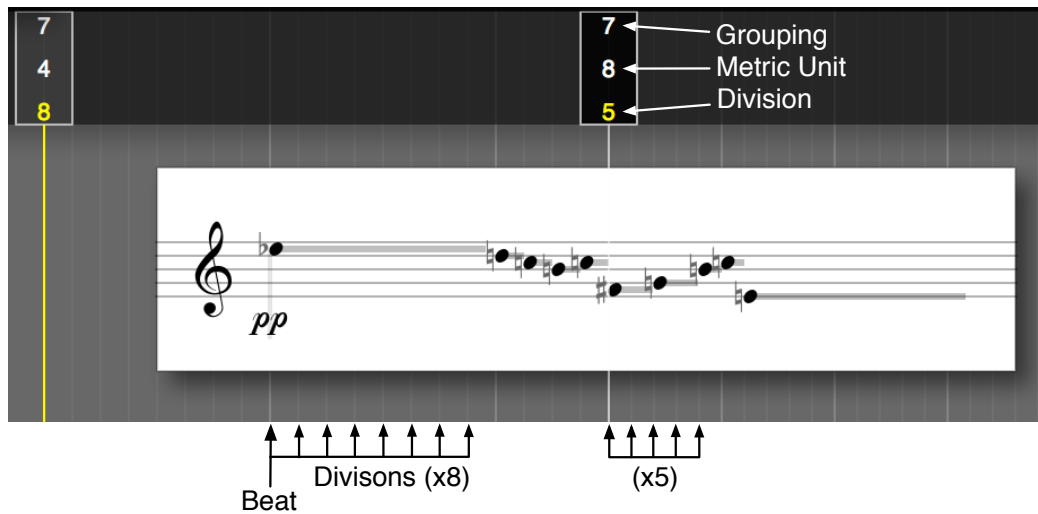


Figure 7.1: Metric Markers in ManuScore.

The method for adding staves in ManuScore was directly inspired by Stravinsky’s self-designed and patented *stylus*, which he used for drawing staves by hand [221]. The stylus allowed him to format manuscript to his needs, and also allowed him to insert additional musical parts into existing scores; borrowing Laske’s term, one could say that his stylus was a product of his “compositional life-cycle.” In ManuScore, staves of arbitrary length can be created at any position in the score, helping to promote the sense of an open musical space. This method of adding staves leads to a “cut-away” style (as seen in Figure 7.2), in which instruments only appear when they are playing [189], creating a visual representation analogous to listening while also supporting the notion of gestalt-based, object-oriented composition that underlies ManuScore’s design.

The image displays a 'cut-away' musical score on a dark grey background. A vertical green line is on the left. Five white rectangular boxes, each representing a musical instrument, are arranged in a staggered, overlapping fashion. Each box contains a staff of musical notation with specific performance markings. The instruments and their markings are: Violin 1 (top left) with 'legato' and 'mp'; Violin 1 (top right) with 'legato'; Violin 2 (middle) with 'ord.' and 'pp'; Viola (middle left) with 'legato', 'mp', and 'pp'; Viola (middle right) with 'p' and 'legato'; and Cello (bottom right) with 'Pizz.' and 'pp'. The notation includes treble clefs for the violins and a bass clef for the cello, with various note values and rests.

Figure 7.2: The “cut-away” score style supports the notion of musical *objects*.

7.2.2 Note Entry in ManuScore.

Once a staff has been created, single notes can be entered in three ways: 1) Typing α and clicking on a staff, 2) Using the Step-Entry Cursor (e key), and 3) Using a MIDI keyboard. It may have been noticed from Figure 7.1 that ManuScore attaches accidentals to all notes, following the practice used by composers like Witold Lutoslawski. In ManuScore, this is a direct result of the inherent lack of conventional bar lines, which traditionally serve to nullify previously written accidentals. Once a note is created, the accidental can be toggled, by holding the COMMAND key and using the ARROW UP/DOWN keys.

Material can also be entered in complete gestures, using the Gesture Tool (g key). This tool allows the user to draw a free-hand line on the staff, which is subsequently interpreted by MusiCog, providing a simple form of gesture interpretation. The Gesture Tool uses MusiCog to infer the pitch contour of the drawn gesture. The sequence of control points in the gesture line is converted to a pitch contour sequence, which is passed to MusiCog for inference at the Schema tier of the pitch model. For each node in the inferred Schema path, the algorithm then selects Invariance and/or Identity tier nodes that best approximate the position of the gesture line. ManuScore's interpretation of a drawn gesture is shown in Figure 7.3. It is worth noting that, although most of the interpreted pitches follow the line quite closely, the (G,F,C) segment (outlined) represents a best-attempt of MusiCog, given its training. MusiCog's failure to follow this segment of the gesture line indicates that, given the inferred musical context, MusiCog did not have a learned representation that could better approximate the path of the line.

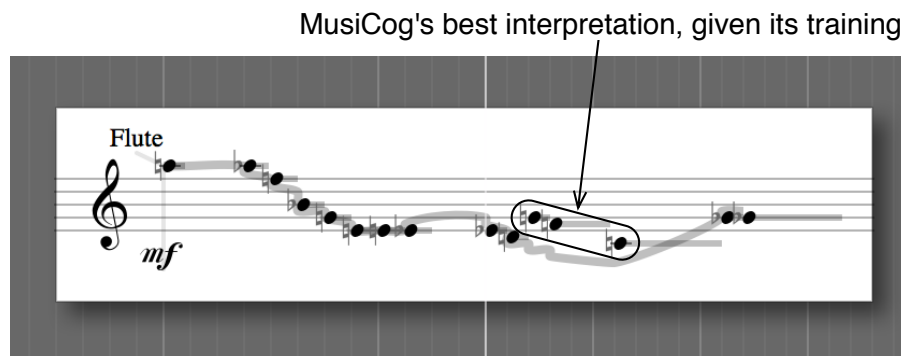


Figure 7.3: MusiCog's interpretation of a Gesture Line.

Finally, new material can also be generated directly by MusiCog, as a continuation of a given musical “seed.” When a note in the seed segment is selected, MusiCog can be made to generate a continuation from that note. In our initial implementation of this function, MusiCog generated a set of possible continuations, rendered the first option to a new staff, and provided a mechanism for toggling through the various options. An example of this approach is shown in Figure 7.4. In the example, the top staff is the user-defined “context” and the lower staff is the generated continuation. The text above the generation “P 10/18 - R 1/4” indicates that the CbCM generated 18 pitch continuations, the 10th of which has been selected, and 4 rhythmic continuations, the 1st of which has been selected. Holding the `OPTION-COMMAND-SHIFT` keys and using the `ARROW UP/DOWN` keys will toggle through the different pitch patterns, while using the `ARROW LEFT/RIGHT` keys will toggle through the different rhythmic patterns. However, working with this implementation could be quite tedious to use, particularly in cases where a large number of pitch and/or rhythm patterns were generated. For this reason, we added an alternative method which could be used to produce a single continuation, rendered directly to the staff containing the musical seed, as shown in Figure 7.5. In this process, the user selects a specific note and uses a contextual menu item to assign the note as the “Generation Trigger.” When the score is played back, MusiCog performs online inference of the music preceding the Generation Trigger note, from which point it begins its continuation. As a result of the inference process, the musical line containing the “trigger” note will be assigned a specific stream (i.e., in MusiCog’s WM), and this stream will be selected for the melodic continuation. As the generation proceeds, output is fed back into the PE (as discussed in Section 5.4), updating MusiCog’s internal state. The continuation proceeds until the musical time corresponding to the end of the Staff passes, at which point the continuation process exits.

7.2.3 Orchestration in ManuScore.

Our goal of maximizing flexibility can also be seen in ManuScore’s approach to orchestration. Rather than following the conventional design, in which instruments are created *a priori*, on “tracks” similar to those used in analog tape recorders, ManuScore uses an approach inspired by the practice of composing to “short-score.” When composers work to short-score, they often add notes about orchestration after the fact, assigning instruments to specific musical gestures directly on the staff. Orchestration in ManuScore follows the same process, as shown in Figure 7.6. If the user’s MIDI system has been configured appropriately, the instrumental switch from Flute to Viola at F \sharp 4 shown in this example will

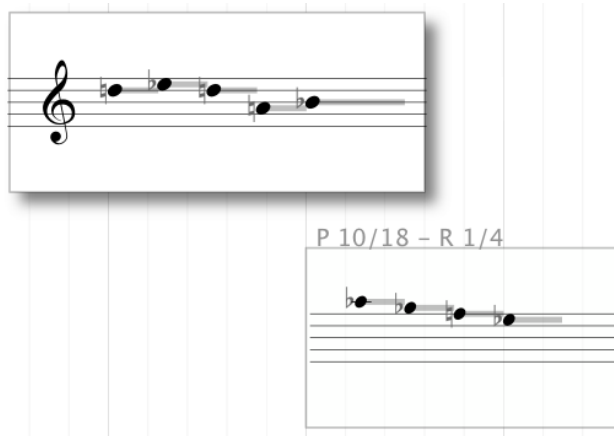


Figure 7.4: MusiCog's presentation of a set of possible continuations from a given musical context. MusiCog offered a set of possible continuations, allowing the user to toggle through the various possibilities.

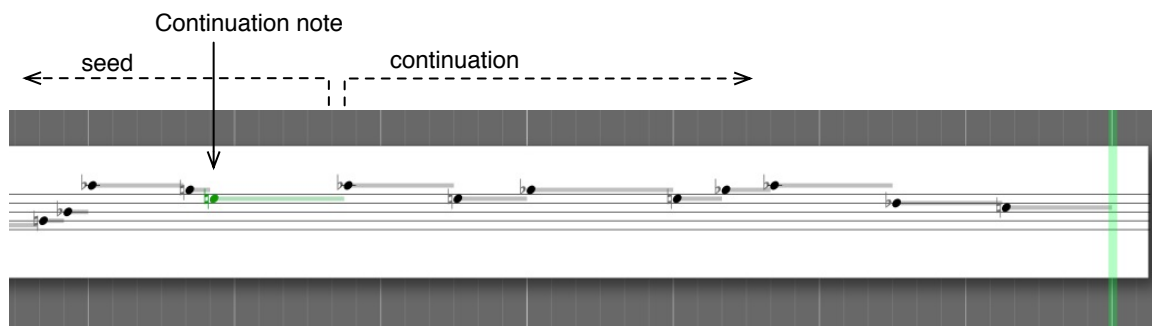


Figure 7.5: The implementation of MusiCog's real time continuation in ManuScore. From a given musical passage a specific "continuation note" is selected. During playback, MusiCog infers the preceding context, then begins continuation from the continuation note.

be played back via MIDI. Instruments can be assigned to notes by typing `i`, clicking at the desired location, and entering the instrument name.

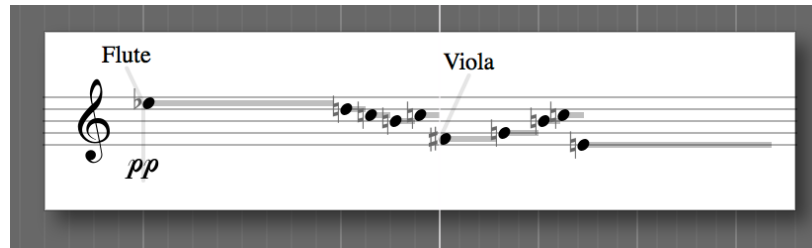


Figure 7.6: Assigning Instruments in ManuScore.

Users can define a custom library of instruments in the MIDI Setup window. The window has four panels: 1) MIDI Setup, 2) Instrument Library, 3) Articulation Library, and 4) Instrument Builder. The MIDI Setup panel allows users to select input/output ports for their MIDI system. In the Instrument Library panel, users can create named Instruments, each with a specific MIDI port and channel assignment. The Articulation Library is used to define named articulation settings—“legato”, “trill minor”, “snap pizz”, etc.—and to configure any program changes, keyswitches, and/or controller changes needed to select these articulations on their MIDI devices. Finally, the Instrument Builder allows users to freely assign Articulations to Instruments. A number of standard, *note-attached* articulations like “staccato”, “down-bow”, “tenuto”, and so on, are assigned to default names in the Articulation Library, and are automatically selected when the appropriate score marking is used. Examples of note-attached articulations can be seen in Figure 7.7. Named articulations (i.e., those not selected directly through notation) can be added to the staff by typing `a`, clicking at the desired point, and entering the Articulation name.

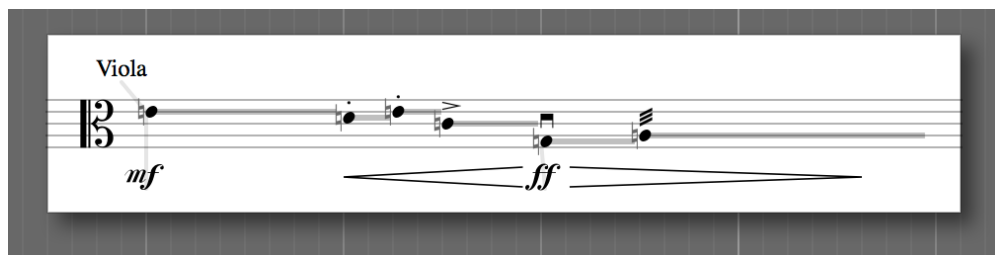


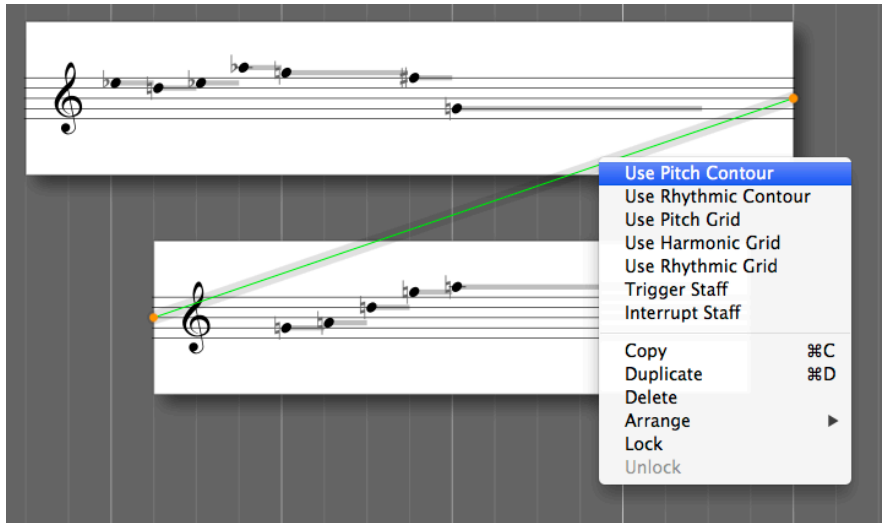
Figure 7.7: Note-attached staccato, accent, down-bow, and tremolo articulations.

7.2.4 Sharing Staff Data with “Links”

A further application of ManuScore’s notion of *objecthood* comes in the form of “links.” A link is a graphical connection between two staves that allows one staff to become a source of information for another—a form of *inheritance*. Typing **1** and clicking on a staff will start the linking process by setting the clicked staff as the “source” of the link. Dragging over another staff and releasing the link will set the staff under the release as the “target” staff. The source staff acts as a data source, and the target staff acts as a receiver of some aspect of the source staff’s data. An example of applying the pitch contour from a source staff to the target staff is shown in Figure 7.8. Link functions currently include the following operations:

- *Pitch contour*: Applies the source staff’s pitch contour to the contents of the target staff. If the target staff has a greater number of events than the source, the source contour is repeated.
- *Rhythmic contour*: Reorders the rhythmic values of events on the target staff to match the rhythmic contour of events on the source staff.
- *Pitch grid*: Provides “crisp” locking of target pitches to source pitches.
- *Harmonic grid*: Provides “fuzzy” locking of target pitches to source pitches. The locking algorithm uses a histogram of pitches used in the score up to the time of the target staff, weighted toward the pitches in the source staff.
- *Rhythmic grid*: Imposes the rhythmic pattern of the source staff onto the contents of the target staff.
- *Trigger Staff*: Allows non-linear playback possibilities by causing the source staff to “trigger” the target staff. When playback of the source staff ends, the target staff begins, regardless of its horizontal position on the score.
- *Interrupt Staff*: If the target staff is playing back at the time when the source staff begins, the target staff is muted; i.e., the source staff “interrupts” the target staff.

Select a Link operation ("Use Pitch Contour")



The operation is performed on the target Staff

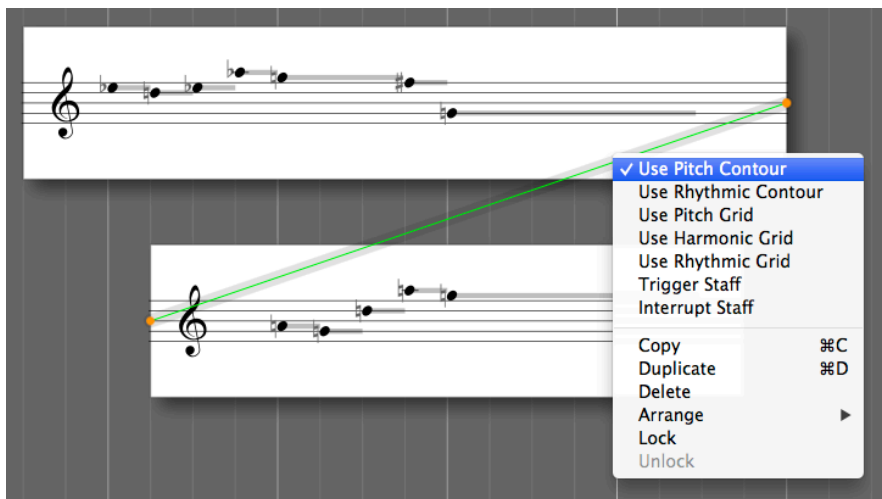


Figure 7.8: Using a *Link* to apply the pitch contour from one staff to another.

7.3 A Composition Study Using ManuScore

In the spring of 2011 we conducted an applied composition study using ManuScore with composer James B. Maxwell, working under the supervision of composer/Professor Owen Underhill. The objective of the study was to test the functionality of the software in a one-to-one composition study context. During the study, Maxwell was to create two short works; one using his regular music notation software package (with which he had been working for many years), and the other using ManuScore. Both pieces were to be approximately 5:00 minutes in duration, and both were to be scored for string quartet. As a further limitation on the process, both works were to draw source material from Fredrick II's "Royal Theme" (best known as the subject of Bach's *Musical Offering*). The two works would be premiered together, in performance, in the fall of 2011, and a listener study conducted at the concert, as described in Section 7.4.

Although the above limitations do not provide a strict enough framework for quantitative evaluation, we do feel that they impose enough commonality on the two compositional processes to isolate, at least to some degree, the software itself as a potential source of difference between the resulting works. Each working process was recorded using video screen capture, in order to provide detailed documentation. An excerpt showing score playback from the ManuScore composition process can be viewed online (audio playback in the clip is directly from ManuScore, using the "Vienna Instruments" software):

http://www.sfu.ca/~jbmaxwel/MusiCog/MnS_experiri.mov.

It is important to note that, at the time of this study, MusiCog was not yet a full-fledged cognitive architecture, but rather contained only a simple melodic segmentation algorithm, and the Closure-based Cueing Model (CbCM)[162]. Generation from the CbCM is carried out in fundamentally the same manner as the CM; inference of a given musical context updates the state of the model, and continuation from that state proceeds via stochastic selection of edges and/or links. Prior to the study, the CbCM was trained on three of Maxwell's prior works: *vovere*, for flute and ensemble, *limina*, for flute, piano, and percussion, and *pensare*, for wind quintet. Although ManuScore is notation-based, it is not a

music notation package, since it does not currently export standard, modern/mensural³ notation. For this reason, a separate process was required to transcribe the completed work into standard music notation for performance. This transcription process revealed some intriguing effects of the ManuScore design, which we discuss further in Section 7.6.

7.4 A Listener Study

The two works composed during the composition study outlined in Section 7.3 were premiered in a concert held at Simon Fraser University's School for the Contemporary Arts, Woodward's campus, in December 2011. The concert was presented by the Musical Metacreations project at SFU, and also included several machine-composed works by composer/Professor Dr. Arne Eigenfeldt, and one other human-composed work, "*One of the above #1*", also by Dr. Eigenfeldt.

Participants in the study were 46 audience members from Vancouver's new music community. The concert featured a total of ten works, written for percussion, string quartet, Disklavier, and other hybrid combinations of these instruments. Each audience member received a concert programme, which explicitly indicated that "machine-composed and machine-assisted musical compositions" would be performed. Each audience member also received an evaluation card on which they were encouraged to provide feedback. On the front side of the evaluation card, audience members were asked to indicate, on a 5-point Likert-scale from 1 to 5, their level of familiarity with contemporary music. This question was followed by ten similar 5-point Likert-scales for rating their level of engagement while listening to each of the compositions. Additionally, audience members were asked to indicate which three pieces they felt were most directly human-composed. All questions on the front side of the card were to be filled out during the performance. The back side of the card contained an additional ten 5-point Likert-scales, asking audience members to indicate the memorability of each piece. This was to be filled out at the end of the concert. However, due to low response rate, this information was excluded from the analysis. Audience members were also given space to write in their own comments. Further details on the study can be found in Eigenfeldt et al. [80].

For the purposes of the current discussion we will focus on the two works composed during the composition study described above (Section 7.3). Since the primary design

³That is, notation grounded on an implicit pulse, in which rhythmic durations are encoded by the graphical appearance of the notes.

goal of ManuScore is to introduce CAC into the compositional process without disrupting the development of a composer's musical language, our working hypothesis was that audience members would not judge the computer-assisted work *experiri* to be implicitly more "human" than the strictly human-composed work, *fundatio*.

7.5 Study Results

In order to avoid the alpha inflation that arises from multiple comparisons, statistical tests were made using *post-hoc* Bonferroni-corrected alpha levels of 0.005 (0.5/10). For part of the analysis, the 46 audience members were divided into *novice* and *expert* groups, based on the score they indicated for the "familiarity with contemporary music" question. The *novice* group consisted of audience members who gave a score of 1-3 out of 5 on the familiarity scale ($N = 25$). The *expert* group consisted of the remaining audience members who gave a 4 or 5 ($N = 19$). Two audience members failed to provide a familiarity score, so their data was excluded from group comparisons.

Table 7.1 gives the engagement rating for all ten works on the programme. The two works composed during the composition study, *fundatio* and *experiri*, are identified in bold type. The score for "*Other, Previously*", also written for string quartet, has been italicized to draw attention to the fact that a highly significant difference between the averaged engagement ratings for all string quartet pieces ($\mu = 4.36, \sigma = 0.73$) and the "*One of the Above*" series of solo percussion pieces ($\mu = 3.36, \sigma = 1.06$) was found, $t(133) = 8.71$; $p < 0.0001$. Similarly, a comparison between the string quartet pieces and the "hybrid" string/percussion pieces *Dead Slow / Look Left* and *Gradual* ($\mu = 3.69, \sigma = 1.09$) was also highly significant, $t(89) = 4.79$; $p < 0.0001$, suggesting that audience members were more engaged by pieces containing strings than by those containing percussion. A comparison between the percussion and hybrid pieces revealed no significant difference, $t(89) = 1.41$; $p = 0.16$ *ns*.

It is worth noting that comparisons between the expert listener engagement ratings for the two works from the composition study, *fundatio* ($\mu = 4.29, \sigma = 0.81$) and *experiri* ($\mu = 4.47, \sigma = 0.61$) were non-significant, $t(18) = 1.00$; $p = 0.33$ *ns*. Novice ratings for *fundatio* ($\mu = 4.24, \sigma = 0.83$) and *experiri* ($\mu = 4.36, \sigma = 0.86$) were similarly non-significant, $t(24) = 0.72$; $p = 0.48$ *ns*. Table 7.2 shows the results for the "directly human-composed" ratings, where it is clear that both *fundatio* and *experiri* were estimated to be human-composed works. Again, there is an effect of instrumentation to be considered, as the

other string quartet work was also highly rated (score in italics). However, there was once again no significant difference between the work composed in ManuScore and the work composed through Maxwell's normal process.

Work	Name & Inst.	Listener Experience		
		Expert	Novice	Comb.
1(c)	<i>In Equilibrio</i> (Disklavier)	3.17 (0.99)	2.71 (1.23)	2.90 (1.14)
2(h)	<i>One of the Above #1</i> (Percussion)	4.00 (1.00)	3.36 (1.19)	3.67 (1.13)
3(c)	<i>Dead Slow/Look Left</i> (Str Qrt & Perc)	4.16 (0.90)	3.08 (1.15)	3.51 (1.16)
4(c)	<i>One of the Above #2</i> (Percussion)	3.68 (0.67)	3.16 (1.07)	3.42 (0.93)
5(h)	<i>fundatio</i> (String Quartet)	4.29 (0.80)	4.24 (0.83)	4.24 (0.81)
6(c-a)	<i>experiri</i> (String Quartet)	4.47 (0.61)	4.36 (0.86)	4.40 (0.76)
7 (c)	<i>One of the Above #3</i> (Percussion)	3.39 (0.76)	3.12 (1.20)	3.22 (1.04)
8 (c)	<i>Other, Previously</i> (String Quartet)	4.31 (0.75)	4.50 (0.59)	4.40 (0.66)
9 (c)	<i>One of the Above #4</i> (Percussion)	3.63 (1.16)	2.71 (1.00)	3.10 (1.16)
10 (c)	<i>Gradual</i> (VI, Perc & Dsk)	4.05 (0.85)	3.88 (0.95)	3.93 (0.89)

Table 7.1: Audience evaluation of “engagement”: (c) computer-composed, (h) human-composed, (c-a) computer-assisted (standard deviations in brackets).

Work	Name	<i>N</i>
1 (c)	<i>In Equilibrio</i>	1
2 (h)	<i>One of the Above #1</i>	12
3 (c)	<i>Dead Slow/Look Left</i>	8
4 (c)	<i>One of the Above #2</i>	2
5 (h)	<i>fundatio</i>	30
6 (c-a)	<i>experiri</i>	27
7 (c)	<i>One of the Above #3</i>	2
8 (c)	<i>Other, Previously</i>	24
9 (c)	<i>One of the Above #4</i>	2
10 (c)	<i>Gradual</i>	14
Total		122

Table 7.2: Evaluation of “directly human-composed”: (c) computer-composed, (h) human-composed, (c-a) computer-assisted (standard deviations in brackets).

7.6 ManuScore as a CAC Tool

One of our primary goals in designing ManuScore was to create an application for composers that would allow them to experiment with interactive, generative, object-oriented composition, in a manner that would not dramatically disrupt their existing musical language. The fact that no significant difference was found in the level of listener “engagement” between *experiri*, composed in ManuScore, and *fundatio* would seem to suggest that this basic goal was achieved. Further, since listeners were not able to identify *experiri* as the computer-assisted work, it appears that the system did not dramatically alter the composer’s musical language. It is also perhaps worth note that, of the two works, the work composed in ManuScore was rated slightly higher in “engagement,” though it is impossible to attribute this preference directly to the influence of ManuScore.

Most of ManuScore’s functionality was utilized during the composition process, though the tools Maxwell reported as most useful were the Gesture Line tool and the “pitch grid” link function. Although Maxwell does consider the final work to be human-composed, he did feel that the software exerted a strong influence on both the form and content of the piece. In particular, he suggested that continuations offered by the CbCM tended to provoke different possibilities for the melodic development of the work, even in cases where the continuations were not included in their original, unedited form. Continuations provided by the CbCM were often edited in both pitch content and rhythm, with the former necessitated by key/scale violations in the generated fragments. It is worth noting that the version

of ManuScore used in the study employed the initial method of presenting continuations to the user, depicted in Figure 7.4. Although Maxwell reported that this method offered a great deal of flexibility, it also became quite tedious to use, particularly in cases where a large number of options were produced and had to be auditioned. Reflection on this experience led to the somewhat more “online” approach added to the final version of ManuScore, as illustrated in Figure 7.5.

Maxwell also reported that working in ManuScore introduced some important changes into the compositional process, which would be worth discussing further. Specifically, the manner in which time is represented, combined with the necessity of transcribing ManuScore documents into standard music notation, should be considered more closely. During the transcription process it was noted that the original rhythmic representation of *experiri* did not always follow an easily interpretable metrical structure. More specifically, it was found that the metrical representation in ManuScore often conflicted with the *implied* metrical structure of the written music, as perceived through listening. An example occurs at the opening of the work, and is shown in Figure 7.9. Looking carefully at the example, we see that the original ManuScore phrase is written using a “beat division” of 5, suggesting a quintuplet pattern. However, it was decided during the transcription process that the perceived structure of the phrase was more appropriately represented using a “4+3” grouping of sixteenth-notes and triplets, rather than the original “5+2” grouping. This change effectively increased the tempo, and shifted the entire metrical structure accordingly.

The figure consists of two parts. The top part shows a snippet of music notation for Violin I in 6/4 time. The notation includes a treble clef, a key signature of one flat, and a dynamic marking of *mp*. The music features a series of sixteenth notes and a triplet of sixteenth notes, followed by a *pp* dynamic marking. The bottom part shows the same music transcribed into standard notation, with a treble clef, a key signature of one flat, and a dynamic marking of *mp*. The transcription includes a *legato* marking and two upward-pointing arrows labeled 'Beat Markers' indicating the start of the first and second beats. Below the transcription, the text 'Beat Division = 5' is written.

Figure 7.9: The opening phrase in ManuScore (bottom) and its transcription into standard music notation.

A similar effect was noticed at measure 12 of the transcription, shown in Figure 7.10. Here we see a passage which was created as a quintuplet pattern in ManuScore (bottom), but transcribed as a grouping of six eighth-notes, under a $\frac{3}{4}$ metre, in standard notation. It was felt that such discrepancies arose primarily as an effect of the purely graphical nature of ManuScore's temporal grid. Since the grid does not impose a specific metrical structure on the music, the cyclical process of writing and listening tended to emphasize perceptual, rather than theoretical, principles in the developing composition. With a temporal grid of 5 beat divisions in place, pitches were easily entered into ManuScore in quintuplet patterns. However, through the iterative process of listening, entering material, editing, and subsequently auditioning edits, the musical form naturally began to unfold according to perceptual/cognitive principles, driven by the musical materials themselves. The phrasing of ideas in the musical foreground gave rise to certain types of groupings, and these naturally gave rise to accompaniments that supported those groupings. Because the metrical representation in ManuScore does not impose any explicit structure on the composed music, the conflict between notation and perception did not become apparent until the transcription process. Further, because the temporal grid was easy to adjust to virtually any beat division value, it was simply not a priority to alter the metrical structure of the score during the composition process. In a sense, quintuplets became naturalized as the "tatum" (the smallest salient rhythmic subdivision of the beat) for the work, and the perceptual reality of the composed rhythmic relationships ultimately superceded their rhythmic representation in ManuScore. It is interesting to consider what would have happened had this work been composed entirely in the conventional notation package, where the use of modern/mensural notation would have imposed strict limitations on the composed music. Presumably, as the first motivic materials explored in the piece, the quintuplet patterns would have remained in place, as would the tempo and metre in which they were originally set, potentially leading to a much different development of the final work. The complete score for *experiri* is included as Appendix A.

The figure shows two representations of the same musical passage for Violin II. The top part is a standard musical score in 3/4 time, marked *mp*. The bottom part is a visualization from ManuScore showing the metric modulation. It features a treble clef staff with notes and rests. Labels include 'Violin 2', 'ord', 'mp', and 'legato'. Below the staff, two upward-pointing arrows are labeled 'Beat Markers', and the text 'Beat Division = 5' is centered below them.

Figure 7.10: The music at measure 12 in ManuScore (bottom), transcribed as a metric modulation in standard music notation.

7.7 Composition in ManuScore with MusiCog

A later compositional study carried out with ManuScore, taking place in the winter of 2012, focused more on autonomous generation, and used the first ManuScore build with a complete version of MusiCog. In this updated version of ManuScore the “online” continuation approach shown in Figure 7.5 was used, however, it did not yet contain the mode/tonality induction and pitch quantization functions. The completed work, *factura*, was performed in May 2012, at Simon Fraser University in Vancouver. This work was intended as a more thorough test of autonomous generation in ManuScore, and thus features extended sections of music generated primarily by MusiCog; in particular, the final movement *factura iii*. The generated output was edited with regard to specific pitch content, dynamics, and articulation, but the contour, interval size, and rhythm of *factura iii* was largely retained from the autonomous generation. The complete score for the work is included as Appendix B.

The composer felt this work to be less successful than *experiri*, largely due to the greater reliance on autonomous machine generation. Accordingly, he found the most interesting parts of the work to be those that were more thoroughly human-composed. However, the work was nevertheless a worthwhile test, as it revealed important limitations of the generative algorithms, leading to subsequent revision of the PM module. The PM implementation in this build of MusiCog relied more heavily on WM contents, and thus had a tendency to produce less motivic variety than the final version discussed in Section 5.4. This can be

seen in the score for *factura iii* (see Appendix B), where much of the material is derived from three simple motives, shown in Figure 7.11. It is also likely that specific limitations imposed by the instrumentation for *factura* (solo percussion versus string quartet) influenced Maxwell's impression of the work's quality, reporting that he found it more challenging to maintain interest in an extended work for solo percussion than for string quartet. Ultimately, however, Maxwell suggested that the greater reliance on autonomous machine composition was the most detrimental factor, suggesting that more of his time was spent on editing than composition.

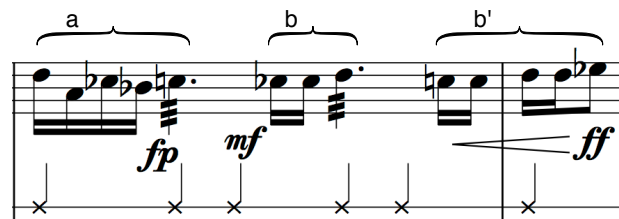


Figure 7.11: Basic motives exploited by MusiCog's autonomous generation process for *factura iii*.

Chapter 8

Conclusion

8.1 Future Work

The implementation of MusiCog presented in Chapter 4 represents a first step. As such, its greatest purpose has been to provide a foundation for future development. Consideration of the problems and challenges encountered along the way has pointed toward some essential improvements to the current model, as well as a number of paths for future development.

8.1.1 Improvements and Extensions to the PE

Although the initial implementation of voice-separation (Section 4.2.1) presented here is fairly sophisticated, we are interested in the possibility of implementing Cambouropoulos' notion of perceptual streams in the PE. This approach is more deeply rooted in auditory stream segregation since, under certain circumstances, it will allow multiple monophonic voices to be joined in a single perceptual stream [32]; e.g., through the perceptual fusion of octave doublings, passages of parallel 3rd or 6ths, and so on. However, as implemented, Cambouropoulos' model presents difficulties for online implementation, as it requires the use of a look-ahead function to decide whether the musical texture is polyphonic or homophonic, which is not practical in an online system like MusiCog. On the other hand, MusiCog's use of association links in the CM does provide a method for representing these kinds of parallel melodic structures, but in MusiCog these are LTM representations, which is inconsistent with the perceptual/cognitive reality supported by Cambouropoulos' model.

It would also be worthwhile to implement a more robust model of tonality induction and tonal composition. Although the current model does provide an estimate of confidence in the induced mode and tonality, it is a valid question whether or not a single tonality should be enforced for an entire generation and, if not, how long a given tonality should last. A more robust approach that models the perception of tonal strength could potentially obviate the need for such a decision, since the choice could be made based on the strength of the induced tonality. In cases where the tonal strength is weak, chromatic alterations of the key could be permitted, or even encouraged. This could lead to a more idiomatic use of chromaticism, particularly in situations where chromatic alterations would serve to increase the tonal strength; for example, during dominant passages in minor keys. On the other hand, it is not clear how such an impulse toward tonal composition would be implemented, and it is unlikely that it could simply be added to the purely stochastic generation model currently in place. It is also worth considering whether a pitch interval Invariance representation is best for tonal contexts, or whether this should be replaced with a scale-step interval representation. For example, the system could use pitch intervals for non-tonal generation, and dynamically switch to scale-step intervals when a tonal centre is desired.

8.1.2 Improvements and Extensions to the PM

Since MusiCog's focus is on the use of an integrated cognitive architecture to model the kind of musical intelligence necessary for composition, much of our attention for future development will be directed toward the PM. Here we outline a number of areas for improvement and development.

Addressing Problems in Rhythmic Generation

We would first like to address problem areas in rhythmic generation discussed in Section 6.3.1. It may be possible to address some of these issues by a simple alteration of the stochastic process used to generate rhythms. Currently, the model attempts to generate rhythmic patterns to match the length of generated pitch patterns. This was done in order to maximize the system's potential for novelty, allowing it to combine pitch patterns and rhythm patterns in novel ways. However, an alternative approach could involve greater exploitation of association links between the pitch and rhythm models. Because both models are trained on segments of events containing pitch and rhythm information, it follows that there will be at least one corresponding rhythmic pattern for every pitch pattern learned. Thus,

during generation, a given sequence of pitch nodes (i.e., a pitch model path) will have a corresponding path in the rhythm model. All that is required is to extract the corresponding rhythm model path for the generated pitch model path. Of course, the disadvantage here is that it will tend to promote quotation from the training corpus.

Another potential solution would be to integrate the CbCM's original linking scheme (outlined in Section 4.2.3 and illustrated in Figure 4.13) into the CM, so that L_2 boundary nodes connect to *both* the L_1 depth $k = 1$ node, representing the note transition immediately following the boundary, and the terminal node, as is currently the case in the CM. This would essentially create a first-order Markov model between the set of L_2 nodes and the set of L_1 depth $k = 1$ nodes, more thoroughly modelling the structure of L_1 segments. Although this would add a considerable amount of linking data to the implementation, it would have the benefit of allowing the PM to incorporate this first-order statistical information into the selection of L_1 paths. Further, such information could also be used for other compositional purposes like constraining the total pitch range, or filtering out uncharacteristically large pitch intervals during generation.

Such an approach would not, however, avoid problems associated with the recombination of different rhythmic subdivisions, as described in Section 6.3.1. It is worth noting that this kind of error is also not generally supported by the music psychological research into rhythmic expectancy, where such patterns would be highly unexpected—and thus unlikely as compositional choices—even for a composer interested in exploring novel rhythmic materials. Therefore, a more cognitively-grounded, motive-level model of rhythmic quantization, offering a similar function to the pitch quantization already implemented in the PE, would likely be sufficient to solve many of these problems. Such a model could operate by dividing the generated rhythmic pattern into beat-length segments, and using rhythmic expectancy to evaluate the well-formedness of the within-beat rhythmic transitions. If a particularly unexpected rhythmic transition was found within the beat, it could be adjusted to conform to the expected rhythmic subdivision. Of course, such a system may be overly aggressive, leading to less interesting and varied rhythmic output, but it would also likely produce more convincing results (i.e., sacrificing novelty for quality and typicality). To reduce the aggressiveness of the system, the choice of whether to apply such quantization could be determined by a beat salience estimation, allowing the mechanism to be shut off in a cognitively-grounded manner when generating non-pulsed music. Such a mechanism would be similar to the use of the “confidence” estimation used to disable pitch quantization, as described in Section 4.2.1 (p. 89).

Closely related to this problem, and implicated in the above solution, is the idea of implementing some form of beat induction in the PE. In practical terms, this would remove the need to provide an external beat, allowing MusiCog to be more easily integrated into live improvisation/composition systems. Further, an implicit sense of pulse would give the system a richer rhythmic knowledge for exploitation during generation. For example, feedback between the PM and PE could be used to control rhythmic tension and release by regulating the PM's tendency to confirm (or deny) the induced pulse.

Polyphonic Generation

It will also be important to move toward polyphonic generation. Given that MusiCog already performs stream/voice separation in the PE, retains multiple streams in WM, and encodes their relationships in the CM, the next logical step is to generate multiple streams in the PM. However, we do feel it is important to first establish a good approach to melodic generation, so that polyphonic generation might benefit from the working melodic model. True polyphonic generation from the CM is a complex problem, as the rhythmic independence and varying segment structure of individual melodic lines in many musical textures will cause independent generated melodies to traverse the CM structure at different rates. Further, empirical investigation into the perception of polyphonic music is limited in the literature [21, 109] (in large part due to methodological difficulties), offering little guidance from a cognitive modelling perspective. One possible approach would be to begin not from the CM hierarchy itself, but rather from the CM's associative memory. For example, a polyphonic generation algorithm could begin by searching for an L_2 node at depth $k = 1$, and from this node could select n strongly associated pitch nodes. For each of the n selected nodes a melodic line could be initiated. As transitions for each line were chosen, the mutual pitch association strength of each candidate node could be tested, and included in the stochastic selection process. Of course, in order to avoid limiting polyphonic generation to chorale-style "block voicings" (i.e., multiple voices with simultaneous onsets) a certain percentage of onsets would have to be permitted to occur asynchronously. Since the rhythmic independence of individual parts in polyphonic textures is an important factor in maintaining the perceptual independence of voices [31, 105], feedback responses of the PE's voice-separation algorithm could, perhaps, be used to guide the selection of melodic continuations in a manner that optimizes voice-separation; another benefit of the integrated approach to generative system design.

High-Level Composition Modelling

Perhaps most importantly, it is essential to devise and implement a more compositionally grounded method for exploiting the considerable potential represented by MusiCog's learned structure and modular design. This suggests a shift in focus from an emphasis on "data models" to the sort of "procedural models" proposed by Laske's composition theory [135] (see Section 2.1). Purely stochastic generation is not likely to rival human composed music, as long as it lacks this sort of procedural model. Generally speaking, probabilistic models have focused on exploiting the statistical information recorded in their data models, rather than on the compositional processes required to create acceptable music from the given data. We suggest this is due, at least in part, to a general misrepresentation of the compositional process underlying stochastic generation. We envision an approach that divides the compositional process into stochastic and deterministic subprocesses, in some ways related to the "pattern based sampling" described by Conklin [43], but also acknowledging certain formal aspects of the topic-elaboration approach proposed by Hayes (Section 2.2). First, stochastic selection will be used to generate an initial set of higher-level plans and low-level motives. These compositional base materials will be fixed as concrete musical concepts, after which the high-level plans will be used to organize the low-level motivic segments in time. Cognitively-grounded, rule-based procedures will adapt the motivic materials to the specific local contexts arising through feedback perception of the developing composition. The WM will serve as a kind of blackboard for these processes and, drawing on ideas from Schmidhuber's theory, will also be used to gauge the subjective cohesiveness and interestingness of the developing composition from the perspective of the agent. These subjective responses could be used to guide the system toward greater or lesser novelty, both at higher levels (e.g., by forcing the generation forward in the formal plan), and at lower levels (e.g., by influencing the rule-based transformation processes; increasing/reducing chromaticism, changing mode/tonality, and so on).

Considered in terms of agent design, a significant limitation of MusiCog is that it lacks a clear method for adapting to its own generative success and/or failure; i.e., it is not a particularly *robust* agent. Drawing inspiration from the ACT-R architecture, we propose to include a "utility" value in the CM, allowing MusiCog to increase the utility of successful compositional choices, and weaken the utility of unsuccessful choices. This could be implemented using an actor-critic RL approach [120], with reward signals generated by the WM module, in consideration of PE responses to feedback input, the state of the WM contents, and the

subjective responses of the agent, as discussed above. Positive reward signals would increase the utility value of CM nodes used in the recent generation (i.e., material currently held in WM as a result of PE feedback), while negative reward signals would lower the utility values. In this way, MusiCog could learn to optimize its own compositional success, without dramatically altering its learned representations, as would happen if reward signals modified the CM edge weights directly.

Finally, because the limitations currently imposed by WM capacity prevent MusiCog from building opus-level compositional structures (i.e., form at the level of complete musical works), and consequently from linking together temporally separated hierarchical substructures in the CM, it would be useful to have a separate mechanism for learning this level of form. We would like to explore a mechanism of cue abstraction, following ideas from Deliège [59]. This system would use simple learning mechanisms to build long time-scale representations of basic formal gestures, or “imprints” [59]. These representations would be sequential, but fundamentally atemporal, so that the specific durations separating imprints would remain unspecified. The imprint representation itself would include abstract analytical information regarding attributes like event density, pitch and rhythmic roughness (proportion of small/short and large/long intervals), chromaticism/modality, and so on. It would also encode basic sequences of similar/contrasting material, by tracking the activation of terminal nodes in the CM during training. This mechanism would not record the terminals themselves, but would rather record ordinal information about the sequences of unique terminals encountered in each training work. This would allow the imprint learning mechanism to encode basic formal patterns analogous to music theoretical notions like “sonata,” “rondo,” and so on, in which different types of materials are assigned ordinal values, and the sequences of ordinals are used to describe the form; e.g., (A,B,A), (A,A,B,B), etc. In this way, the imprint’s abstract analytical information could be used to guide stochastic selection and rule-based transformation processes during the generation of motivic materials, while the ordinal information could be used to build an overall, opus-level formal outline for the generated composition. The abstraction of materials using ordinal values would help protect the selected motives from the sort of continuous variation symptomatic of purely stochastic selection techniques, thus ensuring that the repetition structure and formal organization would remain perceptible in the composed music.

8.1.3 Modelling Attention

MusiCog currently has no explicit model of attention. Although this is a rather serious shortcoming for a proposed cognitive model, it is also an extremely complex subject in the field of music psychology, where only a handful of research projects have attempted to investigate attention direction during polyphonic music listening in natural contexts [87, 94, 109, 203]. Further, due to the experimental challenges involved in identifying the effects of attention in listening subjects, much of the research done has focused either on auditory stream segregation (see Section 3.2), or on neuroimaging studies of the listening brain, which tells us relatively little about the subjective state of the listener. Nevertheless, attention is an important phenomenon to model, since musical form is generally concerned with guiding the listener’s attention through the various stages of musical development, and across the various features of the musical surface. It could be argued that the use of cognitive salience for the retention of items in WM, or the privileging of parallelism as a compositional strategy in the PM, are forms of attention, but these two functions alone certainly do not constitute a sufficient model. A dedicated mechanism for attention will be particularly useful for polyphonic generation, where the system will have to prioritize certain musical parts over others if it is to model the foreground/background relationships found in human music. This is, of course, an area where the integrated approach could be particularly useful, since the system’s own reported attention (i.e., via feedback) could be used to regulate the musical focus of the developing work.

8.2 Final Statements

Although other cognitively-grounded generative models for music exist (see Section 3.6), to the best of our knowledge, MusiCog is the first Integrated Cognitive Architecture for music founded on ideas from the field of ICA design. In this dissertation we have presented musical examples of MusiCog performing perceptual/cognitive tasks including polyphonic voice-separation, melodic segmentation, chunking, hierarchical learning, and melodic generation when trained on corpora of contrasting musical styles. We have performed quantitative evaluations of MusiCog’s proposed “musically naïve” approach to melodic composition, and discussed MusiCog’s output from a music theoretical perspective, with a particular focus on motive/phrase structure. Although MusiCog’s output showed significant differences (i.e., of means) to the training corpora in tests of expectancy-based complexity, entropy, and melodic similarity, style imitation of the more rudimentary source materials (e.g., the

100-song folk corpus) showed great promise. We have shown that the musically naïve approach to generation performs best for music that requires only fundamental perceptual/cognitive processes for comprehension—in particular, scale induction—and that as more abstract music theoretical ideas are implicated in the musical structure of training corpora, MusiCog’s performance decreases. Analysis of musical scores revealed weaknesses of melodic form, perhaps most significantly in the areas of symmetry, repetition structure, and long-term formal development. The system also demonstrated a limited but noteworthy capacity to generate tonal melodies, when trained on tonal material.

The techniques used in MusiCog have been included in the ManuScore CAC environment, which we tested in the context of practical composition studies, the results of which were performed for the general public. This work with ManuScore revealed important strengths and weaknesses of MusiCog in a CAC context, and pointed the way for future development. We have also attempted to show the benefits of an integrated approach to cognitively-grounded music generation. The integrated approach presents intuitively clear solutions to several problems, due in large part to the conceptual symbiosis of cognitive modelling and intellectual reflection and analysis; e.g., if a weakness of the system is revealed during generation, and that weakness is perceptible, then it is reasonable to suggest that the system’s perceptual system might somehow be implicated in the solution. This notion of investigating the relations between the musical behaviour of the system and the states of its various modules—particularly in the context of feedback processing—has only been touched upon. Further investigation is required to develop an in-depth understanding of the potential of this methodological approach.

The relative success of MusiCog’s musically naïve generation process, despite its clear limitations, likely derives from its capacity to model the statistical regularities of music at multiple levels of formal structure; note transitions, motive boundary transitions, and in some cases phrase boundary transitions. However, it is also increasingly apparent that fundamentally stochastic generation, as implemented in MusiCog (and many generative systems), is insufficient for modelling the kind of structure displayed by human-composed music. Even though MusiCog does create detailed representations of such formal regularities, it generally fails to reproduce them during composition, due to its dependence on stochastic selection. It remains to be seen whether Deep Learning approaches will tackle this particular problem. Of course, simply eliminating stochastic processes is not an option for corpus-based systems since, without the influence of stochastic variability, the likelihood of generating verbatim quotations from the corpus increases. This general issue, however,

reveals an interesting problem affecting the development of generative systems; that the concern with achieving novelty generally eclipses the more pressing concern of ensuring quality. Further, it seems that the problems related to lack of repetition structure and/or formal organization observed in the current research, though demonstrated with regard to a limited set of musical styles, are not necessarily exclusive to those styles. That is, these various types of repetition structure and formal organization are not style-specific traits isolated to this music, but are rather basic aspects of musical syntax, conditioned in no small part by the basic perceptual and cognitive capacities of human composers and listeners.

On the other hand, a question that might be worth asking, when reflecting upon MusiCog's difficulties in style imitation, is precisely how many attempts it *should* take for a "composer" to write a good melody? After all, MusiCog's folk generation number 43 was, in many ways, quite acceptable, and did replicate important formal attributes of human-composed melodies. While it's true that it was just one of a hundred attempts, it is also true that it was "composed" in a fraction of a second. So how many attempts at melodic composition should it take to compose a decent melody? How many did it take, for example, to create any one of the songs in our folk corpus? In fact, when considered from the perspective of MusiCog's potential as a CAC tool, it would be reasonable to say that MusiCog is remarkably successful, since, with the push of a button it was able to produce a fairly well-formed, tonal melody that, with only a small amount of editing, could be perfectly convincing. In this light, perhaps "significance" should be considered in a different way? After all, apart from a general understanding that configurational entropy ought to provide for a great number of options for generation from a training corpus, we actually have no way of knowing whether the information contained in a corpus of 100 folk songs should *actually* produce any more than 100 well-formed melodies. It is an assumption. Of course, it seems reasonable to assume that such a corpus does provide the potential for more, but there is no concrete evidence that this is the case.

From the preceding investigation we suggest that what is required for the future development of music generation systems is an implementable theory of composition. Existing theories—with the possible exception of Schmidhuber's theory¹ (see Section 2.5.2)—are purely descriptive (e.g., Collins' Synthesis Process Model [41]), and would require considerable elaboration as prescriptive models to be implemented in computational systems.

¹It is worth noting that MusiCog's use of cognitive salience to regulate the degree of musical parallelism is closely related to Schmidhuber's ideas. However, this approach was shown to be insufficient, since it tends to produce continuously developing patterns of repetition/parallelism and variation, without the tendency to return back to previously exploited material, as is commonly observed in human-composed music.

Nevertheless, in order for high-level, formalizable generative models to move forward, some effort must be made to redress the abandonment of musical knowledge that characterizes many current methods—Pachet’s work with “Markov constraints” [181] is a compelling exception, in this regard. Of course, in order to avoid the pitfalls of the past—producing highly competent systems like EMI, that depend upon inextricably embedded representations of individual musical knowledge—we must be careful to integrate such compositional knowledge in a principled way.

Bibliography

- [1] Online etymology dictionary, URL <http://www.etymonline.com>.
- [2] Abdallah, S. and Plumbley, M., Information dynamics: Patterns of expectation and surprise in the perception of music, *Connection Science*, 21(2-3):89–117, 2009.
- [3] Allan, M., Harmonising chorales in the style of Johann Sebastian Bach, *Master's thesis, School of Informatics, University of Edinburgh*, 2002.
- [4] Allauzen, C., Crochemore, M., and Raffinot, M., Factor oracle: A new structure for pattern matching, in *SOFSEM'99: Theory and Practice of Informatics*, pp. 295–310, Springer, 1999.
- [5] Amabile, T. M., *Creativity in context: Update to "the social psychology of creativity."*, Westview press, 1996.
- [6] Ames, C., The Markov process as a compositional model: a survey and tutorial, *Leonardo*, 22(2):175–187, 1989, ISSN 0024-094X.
- [7] Anderson, J., ACT: A simple theory of complex cognition, *American Psychologist*, 51:355–365, 1996, ISSN 0003-066X.
- [8] Andrews, M. W. and Dowling, W. J., The development of perception of interleaved melodies and control of auditory attention, *Music Perception*, pp. 349–368, 1991.
- [9] Assayag, G., Bloch, G., Chemillier, M., Cont, A., and Dubnov, S., Omax brothers: a dynamic topology of agents for improvisation learning, in *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pp. 125–132, ACM, 2006.
- [10] Assayag, G., Rueda, C., Laurson, M., Agon, C., and Delerue, O., Computer-assisted composition at IRCAM: from Patchwork to OpenMusic, *Computer Music Journal*, 23(3):59–72, 1999.
- [11] Aucouturier, J.-J. and Pachet, F., Music similarity measures: What's the use, in *Proc. ISMIR*, vol. 2, 2002.
- [12] Baddeley, A., Working memory, *Science*, 255(5044):556–559, 1992.
- [13] Baddeley, A., Working memory: Looking back and looking forward, *Nature Reviews Neuroscience*, 4(10):829–839, 2003.

- [14] Baroni, M., Musical grammar and the study of cognitive processes of composition, *Musicæ Scientiæ*, 3(1):3–21, 1999.
- [15] Bel, B., Kippen, J. et al., Modelling music with grammars: formal language representation in the Bol Processor, *Computer Representations and Models in Music*, pp. 207–238, 1992.
- [16] Bengio, Y., Learning deep architectures for AI, *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.
- [17] Bengtsson, S. L., Ullén, F., Henrik Ehrsson, H., Hashimoto, T., Kito, T., Naito, E., Forssberg, H., and Sadato, N., Listening to rhythms activates motor and premotor cortices, *Cortex*, 45(1):62–71, 2009.
- [18] Bernstein, L., *The unanswered question: Six talks at Harvard*, Harvard University Press, 1976.
- [19] Berz, W. L., Working memory in music: A theoretical model, *Music Perception*, pp. 353–364, 1995.
- [20] Bickerman, G., Bosley, S., Swire, P., and Keller, R., Learning to create jazz melodies using Deep Belief Nets, in *Proceedings Of The International Conference On Computational Creativity, Lisbon, Portugal*, 2010.
- [21] Bigand, E., McAdams, S., and Forêt, S., Divided attention in music, *International Journal of Psychology*, 35(6):270–278, 2000.
- [22] Bizzell, P. and Herzberg, B., *The rhetorical tradition*, Bedford Books of St. Martin's Press Boston, 1990.
- [23] Blair, R. C. and Karniski, W., An alternative method for significance testing of waveform difference potentials, *Psychophysiology*, 30(5):518–524, 1993.
- [24] Boden, M. A., Creativity and artificial intelligence, *Artificial Intelligence*, 103(1):347–356, 1998.
- [25] Boltz, M., Some structural determinants of melody recall, *Memory & Cognition*, 19(3):239–251, 1991.
- [26] Bourdin, B. and Fayol, M., Is written language production more difficult than oral language production? a working memory approach, *International Journal of Psychology*, 29(5):591–620, 1994.
- [27] Brower, C., A cognitive theory of musical meaning, *Journal of music theory*, 44(2):323–379, 2000.
- [28] Brown, A., How the computer assists composers: A survey of contemporary practise, 2001.
- [29] Brown, A., Modes of compositional engagement, *Mikropolyphony*, 6, 2001.

- [30] Brown, G. J. and Cooke, M., Perceptual grouping of musical sounds: A computational model, *Journal of New Music Research*, 23(2):107–132, 1994.
- [31] Cambouropoulos, E., Musical parallelism and melodic segmentation:: A computational approach, *Music Perception*, 23(3):249–268, 2006.
- [32] Cambouropoulos, E., Voice and stream: Perceptual and computational modeling of voice separation, *Music Perception*, 26(1):75–94, 2008.
- [33] Carlsen, J. C., Divenyi, P. I., and Taylor, J. A., A preliminary study of perceptual expectancy in melodic configurations, *Bulletin of the Council for Research in Music Education*, pp. 4–12, 1970.
- [34] Chadabe, J., Interactive composing: an overview, *Computer Music Journal*, 8(1):22–27, 1984, ISSN 0148-9267.
- [35] Chaffin, R. and Imreh, G., Practicing perfection: Piano performance as expert memory, *Psychological Science*, 13(4):342–349, 2002.
- [36] Chew, E. and Wu, X., Separating voices in polyphonic music: A contig mapping approach, in *Computer Music Modeling and Retrieval*, pp. 1–20, Springer, 2005.
- [37] Chikhaoui, B., Pigot, H., Beaudoin, M., Pratte, G., Bellefeuille, P., and Laudares, F., Learning a song: an ACT-R model.
- [38] Chomsky, N. and DiNozzi, R., *Language and mind*, Harcourt Brace Jovanovich New York, 1972.
- [39] Chong, H., Tan, A., and Ng, G., Integrated cognitive architectures: a survey, *Artificial Intelligence Review*, 28(2):103–130, 2007, ISSN 0269-2821.
- [40] Clementi and Company, *The Melographicon: a New Musical Work, by which an Interminable Number of Melodies May be Produced: And Young People who Have a Taste for Poetry Enabled to Set Their Verses to Music for the Voice and Piano-forte, Without the Necessity of a Scientific Knowledge of the Art ...*, Clementi and Company, 1805, URL <http://books.google.ca/books?id=Y-AsAAAAYAAJ>.
- [41] Collins, D., A synthesis process model of creative thinking in music composition, *Psychology of Music*, 33(2):193, 2005.
- [42] Collins, T., *Improved methods for pattern discovery in music, with applications in automated stylistic composition*, Ph.D. thesis, The Open University, 2011.
- [43] Conklin, D., Music generation from statistical models, in *Proceedings of the AISB 2003 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, pp. 30–35, Citeseer, 2003.
- [44] Conklin, D. and Bergeron, M., Feature set patterns in music, *Computer Music Journal*, 32(1):60–70, 2008.

- [45] Conklin, D. and Witten, I. H., Multiple viewpoint systems for music prediction, *Journal of New Music Research*, 24(1):51–73, 1995.
- [46] Cont, A., Dubnov, S., and Assayag, G., Anticipatory model of musical style imitation using collaborative and competitive reinforcement learning, in *Anticipatory behavior in adaptive learning systems*, pp. 285–306, Springer, 2007.
- [47] Cope, D., An expert system for computer-assisted composition, *Computer Music Journal*, 11(4):30–46, 1987.
- [48] Cope, D., Recombinant music: using the computer to explore musical style, *Computer*, 24(7):22–28, 1991.
- [49] Cope, D., Computer modeling of musical intelligence in EMI, *Computer Music Journal*, 16(2):69–83, 1992.
- [50] Cope, D., The Composer's Underscoring Environment: CUE, *Computer Music Journal*, 21(3):20–37, 1997.
- [51] Cope, D., *Virtual Music: computer synthesis of musical style*, MIT Press, Cambridge, MA, USA, 2001.
- [52] Cope, D., *Computer Models of Musical Creativity*, MIT Press, Cambridge, MA, USA, 2005.
- [53] Cowan, N., The magical mystery four how is working memory capacity limited, and why?, *Current Directions in Psychological Science*, 19(1):51–57, 2010.
- [54] Cruz-Alcázar, P. P. and Vidal-Ruiz, E., Learning regular grammars to model musical style: Comparing different coding schemes, in *Grammatical Inference*, pp. 211–222, Springer, 1998.
- [55] Csikszentmihalyi, M., *Reflections on the field.*, 1998.
- [56] De Cheveigne, A., Pitch perception models, in *Pitch*, pp. 169–233, Springer, 2005.
- [57] Deliège, I., Mechanisms of cue extraction in memory for musical time, *Contemporary Music Review*, 9(1-2):191–205, 1993.
- [58] Deliège, I., Cue abstraction as a component of categorisation processes in music listening, *Psychology of Music*, 24(2):131–156, 1996.
- [59] Deliège, I., Prototype effects in music listening: An empirical approach to the notion of imprint, *Music Perception*, 18(3):371–407, 2001.
- [60] Desain, P., A (de)composable theory of rhythm perception, *Music Perception*, 9(4):439–454, 1992, ISSN 0730-7829.
- [61] Desain, P., Honing, H. et al., The formation of rhythmic categories and metric priming, *PERCEPTION-LONDON*, 32(3):341–366, 2003.

- [62] Deutsch, D., *The psychology of music*, Academic Pr, 1999, ISBN 0122135652.
- [63] Deutsch, D., *The psychology of music*, Academic Press, 2012.
- [64] Deutsch, D. and Feroe, J., The internal representation of pitch sequences in tonal music., *Psychological Review*, 88(6):503, 1981, ISSN 1939-1471.
- [65] Dictionary, O. E., Oxford english dictionary online, URL <http://www.oxforddictionaries.com>.
- [66] Dillmann, R. and Asfour, T., Deliverable title: State of the art in intelligent and cognitive systems.
- [67] DiScipio, A., Formalization and intuition in Analogique A et B, in *Proceedings of the international symposium iannis xenakis*, pp. 95–108, 2005.
- [68] Dowling, W., Scale and contour: Two components of a theory of memory for melodies, *Psychological Review*, 85(4):341–354, 1978, ISSN 0033-295X.
- [69] Dowling, W., Context effects on melody recognition: Scale-step versus interval representations, *Music Perception*, 3(3):281–296, 1986, ISSN 0730-7829.
- [70] Dowling, W., *The development of music perception and cognition*, Foundations of Cognitive Psychology. Cambridge: MIT Press, 1999.
- [71] Dowling, W. and Bartlett, J., The importance of interval information in longterm memory for melodies, *Psychomusicology: Music, Mind and Brain*, 1(1), 2008.
- [72] Dowling, W. J. and Harwood, D. L., *Music cognition*, vol. 19986, Academic Press New York, 1986.
- [73] Dubnov, S. and Assayag, G., Universal prediction applied to stylistic music generation, in *Mathematics and Music, A Diderot Mathematical Forum*, Assayag, G.; Feichtinger, HG, pp. 147–160, 2002.
- [74] Duch, W., Oentaryo, R., and Pasquier, M., Cognitive architectures: Where do we go from here?, in *Proceeding of the 2008 conference on Artificial General Intelligence 2008: Proceedings of the First AGI Conference*, pp. 122–136, IOS Press, 2008.
- [75] Duncker, K. and Lees, L. S., On problem-solving., *Psychological monographs*, 58(5):i, 1945.
- [76] Edworthy, J., Interval and contour in melody processing, *Music Perception*, 2(3):375–388, 1985, ISSN 0730-7829.
- [77] Eerola, T. and North, A. C., Expectancy-based model of melodic complexity, in *Proceedings of the Sixth International Conference on Music Perception and Cognition. Keele, Staffordshire, UK: Department of Psychology. CD-ROM*, 2000.
- [78] Eerola, T. and Toiviainen, P., MIR in Matlab: The MIDI Toolbox., in *ISMIR*, 2004.

- [79] Eigenfeldt, A., The evolution of evolutionary software: intelligent rhythm generation in Kinetic Engine, in *Applications of Evolutionary Computing*, pp. 498–507, Springer, 2009.
- [80] Eigenfeldt, A., Burnett, A., and Pasquier, P., Evaluating musical metacreation in a live performance context, in *Proceedings of the Third International Conference on Computational Creativity*, pp. 140–144, 2012.
- [81] Eigenfeldt, A. and Pasquier, P., Considering vertical and horizontal context in corpus-based generative electronic dance music, in *Proceedings of the Fourth International Conference on Computational Creativity*, p. 72, 2013.
- [82] Elsea, P., Fuzzy logic and musical decisions, *University of California, Santa Cruz*, 1995.
- [83] Evers, T. and Musse, S., Building artificial memory to autonomous agents using dynamic and hierarchical finite state machine, in *Computer Animation, 2002. Proceedings of*, pp. 164–169, IEEE, 2002, ISBN 0769515940.
- [84] Feldman, J. A. and Ballard, D. H., Connectionist models and their properties, *Cognitive science*, 6(3):205–254, 1982.
- [85] Flower, L. and Hayes, J. R., A cognitive process theory of writing, *College composition and communication*, 32(4):365–387, 1981.
- [86] Fuegi, J. and Francis, J., Lovelace & babbage and the creation of the 1843 'notes', *Annals of the History of Computing, IEEE*, 25(4):16–26, 2003.
- [87] Fujioka, T., Trainor, L. J., Ross, B., Kakigi, R., and Pantev, C., Automatic encoding of polyphonic melodies in musicians and non-musicians, *Journal of cognitive neuroscience*, 17(10):1578–1592, 2005.
- [88] Fuller, F. D., *Development of topic-comment algorithms and text structures in written compositions of students in grades one through nine*, Ph.D. thesis, University of Washington, 1995.
- [89] Gillick, J., Tang, K., and Keller, R. M., Machine learning of jazz grammars, *Computer Music Journal*, 34(3):56–66, 2010.
- [90] Ginsborg, J., Strategies for memorizing music, *Musical excellence: strategies and techniques to enhance performance*, pp. 123–141, 2004.
- [91] Godoy, R., Jensenius, A., and Nymoer, K., Chunking in music by coarticulation, *Acta Acustica united with Acustica*, 96(4):690–700, 2010.
- [92] Gonzalez Thomas, N., Pasquier, P., Eigenfeldt, A., and Maxwell, J. B., A methodology for the comparison of melodic generation models using Meta-Melo., in *Proceedings of the 13th International Society for Music Information Retrieval Conference*, Curitiba, PR, Brazil, 2013.

- [93] Gonzalez Thomas, N., Pasquier, P., Loughin, T., and Burnet, A., In publication.
- [94] Gregory, A. H., Listening to polyphonic music, *Psychology of Music*, 18(2):163–170, 1990.
- [95] Groppe, D., *One sample/paired sample permutation t-test with correction for multiple comparisons*, MATLAB Central File Exchange, <http://www.mathworks.com/matlabcentral/fileexchange/29782>, Retrieved Jan 7, 2014.
- [96] Gruber, H. E. and Barrett, P. H., *Darwin on man: A psychological study of scientific creativity.*, EP Dutton, 1974.
- [97] Hawkins, J., *On intelligence*, Macmillan, 2004.
- [98] Hawkins, J., George, D., and Niemasik, J., Sequence memory for prediction, inference and behaviour, *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1521):1203–1209, 2009.
- [99] Hayes, J. R., Modeling and remodeling writing, *Written Communication*, 29(3):369–388, 2012.
- [100] Heider, F., Gestalt theory: Early history and reminiscences., *Journal of the History of the Behavioral Sciences*, 1970.
- [101] Heller-Roazen, D., *The Fifth Hammer: Pythagoras and the Disharmony of the World*, Zone Books, 2011.
- [102] Henson, R. N., Short-term memory for serial order: The start-end model, *Cognitive psychology*, 36(2):73–137, 1998.
- [103] Herholz, S. C., Halpern, A. R., and Zatorre, R. J., Neuronal correlates of perception, imagery, and memory for familiar tunes, *Journal of cognitive neuroscience*, 24(6):1382–1397, 2012.
- [104] Horner, A. and Goldberg, D., Genetic algorithms and computer-assisted music composition, *Urbana*, 51(61801):14, 1991.
- [105] Huron, D., Tone and voice: A derivation of the rules of voice-leading from perceptual principles, *Music Perception*, 19(1):1–64, 2001.
- [106] Huron, D. and Parncutt, R., An improved model of tonality perception incorporating pitch salience and echoic memory, *Psychomusicology: Music, Mind & Brain*, 12(2):154–171, 1993.
- [107] Huron, D. and Royal, M., What is melodic accent? converging evidence from musical practice, *Music Perception*, pp. 489–516, 1996.

- [108] Huron, D. B., *Sweet anticipation : music and the psychology of expectation*, MIT Press, Cambridge, Mass., 2006, 2005054013 GBA635835 David Huron. ill., music ; 24 cm. "A Bradford book." Includes bibliographical references (p. [423]-448) and index.
- [109] Janata, P., Tillmann, B., and Bharucha, J. J., Listening to polyphonic music recruits domain-general attention and working memory circuits, *Cognitive, Affective, & Behavioral Neuroscience*, 2(2):121–140, 2002.
- [110] Johanson, B. and Poli, R., *GP-music: An interactive genetic programming system for music generation with automated fitness raters*, Citeseer, 1998.
- [111] Jordanous, A., Voice separation in polyphonic music: A data-driven approach, in *Proceedings of the International Computer Music Conference*, 2008.
- [112] Karim, S., *Acquiring Plans Within Situated Resource-bounded Agents: A Hybrid BDI-based Approach*, Ph.D. thesis, University of Melbourne, Dept. of Information Systems, 2009.
- [113] Karydis, I., Nanopoulos, A., Papadopoulos, A., Cambouropoulos, E., and Manolopoulos, Y., Horizontal and vertical integration/segregation in auditory streaming: a voice separation algorithm for symbolic musical data, in *Proceedings 4th Sound and Music Computing Conference (SMC'2007)*, 2007.
- [114] Keller, R. and Morrison, D. R., A grammatical approach to automatic improvisation, in *Proceedings, Fourth Sound and Music Conference, Lefkada, Greece, July.* "Most of the soloists at Birdland had to wait for Parker's next record in order to find out what to play next. What will they do now, 2007.
- [115] Kidd, G., Boltz, M., and Jones, M. R., Some effects of rhythmic context on melody recognition, *The American journal of psychology*, pp. 153–173, 1984.
- [116] Kilian, J. and Hoos, H. H., Voice separation-a local optimization approach., in *ISMIR*, 2002.
- [117] Koelsch, S., Toward a neural basis of music perception—a review and updated model, *Frontiers in psychology*, 2, 2011.
- [118] Koelsch, S., Schröger, E., and Tervaniemi, M., Superior pre-attentive auditory processing in musicians, *Neuroreport*, 10(6):1309–1313, 1999.
- [119] Kohonen, T., A self-learning musical grammar, or 'associative memory of the second kind', in *Neural Networks, 1989. IJCNN., International Joint Conference on*, pp. 1–5, IEEE, 1989.
- [120] Konda, V. R. and Tsitsiklis, J. N., Actor-critic algorithms., in *NIPS*, vol. 13, pp. 1008–1014, Citeseer, 1999.

- [121] Koniari, D., Predazzer, S., and Mélen, M., Categorization and schematization processes used in music perception by 10-to 11-year-old children, *Music Perception*, 18(3):297–324, 2001.
- [122] Kozbelt, A., Beghetto, R. A., and Runco, M. A., Theories of creativity, *The Cambridge handbook of creativity*, pp. 20–47, 2010.
- [123] Krumhansl, C. L., Effects of musical context on similarity and expectancy, *Systematische musikwissenschaft*, 3(2):211–250, 1995.
- [124] Krumhansl, C. L., Music psychology and music theory: Problems and prospects, *Music Theory Spectrum*, pp. 53–80, 1995.
- [125] Krumhansl, C. L. and Kessler, E. J., Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys., *Psychological review*, 89(4):334, 1982.
- [126] Krumhansl, C. L. and Schmuckler, M. A., Key-finding in music: An algorithm based on pattern matching to tonal hierarchies, in *Mathematical Psychology Meeting*, 1986.
- [127] Lamont, A. and Dibben, N., Motivic structure and the perception of similarity, *Music Perception*, 18(3):245–274, 2001, ISSN 0730-7829.
- [128] Lang, P. H., The enlightenment and music, *Eighteenth-Century Studies*, 1(1):93–108, 1967.
- [129] Langley, P. and Choi, D., A unified cognitive architecture for physical agents, in *Proceedings of the National Conference on Artificial Intelligence*, vol. 21, p. 1469, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.
- [130] Langley, P., Laird, J., and Rogers, S., Cognitive architectures: Research issues and challenges, *Cognitive Systems Research*, 10(2):141–160, 2009, ISSN 1389-0417.
- [131] Lartillot, O., A musical pattern discovery system founded on a modeling of listening strategies, *Computer Music Journal*, 28(3):53–67, 2004, ISSN 0148-9267.
- [132] Lartillot, O. and Toiviainen, P., Motivic matching strategies for automated pattern extraction, *Musicae Scientiae*, 11(1 suppl):281, 2007.
- [133] Lashley, K. S., The problem of serial order in behavior, 1951, pp. 112–135, 1951.
- [134] Laske, O., Composition theory in Koenig's Project One and Project Two, *Computer Music Journal*, 5(4):54–65, 1981, ISSN 0148-9267.
- [135] Laske, O., Composition theory: An enrichment of music theory, *Journal of New Music Research*, 18(1):45–59, 1989, ISSN 0929-8215.
- [136] Laske, O., The computer as the artist's alter ego, *Leonardo*, pp. 53–66, 1990.
- [137] Laske, O., Toward an epistemology of composition, *Journal of new music research*, 20(3):235–269, 1991, ISSN 0929-8215.

- [138] Laske, O., Mindware and software: Can they meet?: Observations on AI and the arts, in *IAKTA/LIST International Workshop on Knowledge Technology in the Arts*, 1993.
- [139] Laurson, M., Kuuskankare, M., and Norilo, V., An overview of PWGL, a visual programming environment for music, *Computer Music Journal*, 33(1):19–31, 2009.
- [140] Leach, J. and Fitch, J., Nature, music, and algorithmic composition, *Computer Music Journal*, 19(2):23–33, 1995, ISSN 0148-9267.
- [141] Lebiere, C. and Wallach, D., Sequence learning in the ACT-R cognitive architecture: Empirical analysis of a hybrid model, *Sequence learning*, pp. 188–212, 2001.
- [142] Lebiere, C., Wallach, D., and Taatgen, N., Implicit and explicit learning in ACT-R, 1998.
- [143] Legendre, P. and Legendre, L., *Numerical ecology*, vol. 20, Elsevier, 2012.
- [144] Lehman, J., Laird, J., and Rosenbloom, P., A gentle introduction to Soar, an architecture for human cognition, *Invitation to cognitive science*, 4:212–249, 1996.
- [145] Leman, M., Lesaffre, M., and Tanghe, K., Introduction to the IPEM toolbox for perception-based music analysis, *Mikropolyphonie-The Online Contemporary Music Journal*, 7, 2001.
- [146] Lerdahl, F., Jackendoff, R., and Jackendoff, R., *A generative theory of tonal music*, The MIT Press, 1996, ISBN 026262107X.
- [147] Levelt, W. J. and Barnas, A., *Formal grammars in linguistics and psycholinguistics*, John Benjamins Publishing Company, 1974.
- [148] Levitin, D., Memory for musical attributes, *Foundations of cognitive psychology: Core readings*, pp. 295–310, 2002.
- [149] Logan, G., Toward an instance theory of automatization, *Psychological review*, 95(4):492–527, 1988, ISSN 0033-295X.
- [150] Logie, R. H., *Visuo-spatial working memory*, Psychology Press, 1995.
- [151] Loui, P., Wessel, D., and Kam, C., Acquiring new musical grammars: a statistical learning approach, in *28th Annual Conference of the Cognitive Science Society*, pp. 1711–1716.
- [152] Loui, P., Wessel, D. L., and Kam, C. L. H., Humans rapidly learn grammatical structure in a new musical scale, *Music perception*, 27(5):377, 2010.
- [153] Loula, A., Gudwin, R., and Queiroz, J., *Artificial cognition systems*, IGI Global, 2007, ISBN 1599041111.

- [154] MacDonald, R., Byrne, C., and Carlton, L., Creativity and flow in musical composition: an empirical investigation, *Psychology of Music*, 34(3):292, 2006, ISSN 0305-7356.
- [155] Mackintosh, N., Neurobiology, psychology and habituation, *Behaviour Research and Therapy*, 25(2):81–97, 1987.
- [156] Madsen, S. T. and Widmer, G., Separating voices in MIDI., in *ISMIR*, pp. 57–60, 2006.
- [157] Manaris, B., Roos, P., Machado, P., Krehbiel, D., Pellicoro, L., and Romero, J., A corpus-based hybrid approach to music analysis and composition, in *Proceedings of the National Conference on Artificial Intelligence*, vol. 22, p. 839, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.
- [158] Margulis, E., A model of melodic expectation, *Music Perception*, 22(4):663–713, 2005, ISSN 0730-7829.
- [159] Maxwell, J. and Eigenfeldt, A., The MusicDB: A music database query system for recombination-based composition in Max/MSP, in *Proceedings of the 2008 International Computer Music Conference*, 2008.
- [160] Maxwell, J., Eigenfeldt, A., Pasquier, P., and Thomas, N., MusiCog: A cognitive architecture for music learning and generation, in *Proceedings of the 2012 Sound and Music Computing Conference*, 2012.
- [161] Maxwell, J., Pasquier, P., and Eigenfeldt, A., Hierarchical Sequential Memory for Music: A cognitive model., in *ISMIR*, pp. 429–434, 2009.
- [162] Maxwell, J., Pasquier, P., and Eigenfeldt, A., The Closure-based Cueing Model: Cognitively-inspired learning and generation of musical sequences, in *Proceedings of the 2011 Sound and Music Computing Conference*, 2011.
- [163] Maxwell, J. B., Eigenfeldt, A., and Pasquier, P., *ManuScore: Music notation-based computer-assisted composition*, Proceedings of the 2012 International Computer Music Conference, 2012.
- [164] McAdams, S., Audition: Cognitive psychology of music, *The Mind-Brain Continuum*, pp. 251–279, 1996.
- [165] McCormack, J., McIlwain, P., Lane, A., and Dorin, A., Generative composition with Nodal, in *Workshop on Music and Artificial Life (part of ECAL 2007)*, Lisbon, Portugal, Citeseer, 2007.
- [166] McCutchen, D., A capacity theory of writing: Working memory in composition, *Educational Psychology Review*, 8(3):299–325, 1996.
- [167] McCutchen, D., Knowledge, processing, and working memory: Implications for a theory of writing, *Educational Psychologist*, 35(1):13–23, 2000.

- [168] McCutchen, D., From novice to expert: Implications of language skills and writing-relevant knowledge for memory during the development of writing skill, *Journal of Writing Research*, 3(1):51–68, 2011.
- [169] Mishra, J., Effects of structure and serial position on memory errors in musical performance, *Psychology of Music*, 38(4):447–461, 2010.
- [170] Mitchell, M., An introduction to genetic algorithms (complex adaptive systems), 1998.
- [171] Morris, R., Tarassenko, L., and Kenward, M., *Cognitive systems: information processing meets brain science*, Academic Press, 2006, ISBN 0120885662.
- [172] Müllensiefen, D. and Frieler, K., Cognitive adequacy in the measurement of melodic similarity: Algorithmic vs. human judgments, *Computing in Musicology*, 13(2003):147–176, 2004.
- [173] Müllensiefen, D. and Frieler, K., The simile algorithms documentation 0.3, *White Paper*, 2006.
- [174] Narmour, E., The “genetic code” of melody: Cognitive structures generated by the implication-realization model, *Contemporary Music Review*, 4(1):45–63, 1989.
- [175] Narmour, E. and of Chicago, U., *The analysis and cognition of basic melodic structures: The implication-realization model*, University of Chicago Press, 1990, ISBN 0226568458.
- [176] Newell, A., Simon, H. A. et al., *Human problem solving*, vol. 14, Prentice-Hall Englewood Cliffs, NJ, 1972.
- [177] Nierhaus, G., *Algorithmic composition*, Springer, 2009.
- [178] Ockelford, A., On similarity, derivation and the cognition of musical structure, *Psychology of Music*, 32(1):23–74, 2004.
- [179] Ockelford, A., Another exceptional musical memory: evidence from a savant of how atonal music is processed in cognition, *Music and the Mind: Essays in Honour of John Sloboda*, p. 237, 2011.
- [180] Pachet, F., The Continuator: Musical interaction with style, *Journal of New Music Research*, 32(3):333–341, 2003.
- [181] Pachet, F., Roy, P., and Barbieri, G., Finite-length Markov processes with constraints, in *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One*, pp. 635–642, AAAI Press, 2011.
- [182] Parncutt, R., Template-matching models of musical pitch and rhythm perception, *Journal of New Music Research*, 23(2):145–167, 1994.
- [183] Patel, A. D., *Music, language, and the brain*, Oxford University Press, USA, 2010.

- [184] Pearce, M., Müllensiefen, D., and Wiggins, G., A comparison of statistical and rule-based models of melodic segmentation, in *Proceedings of the Ninth International Conference on Music Information Retrieval*, pp. 89–94, 2008.
- [185] Pearce, M. T., Müllensiefen, D., and Wiggins, G. A., Melodic grouping in music information retrieval: New methods and applications, in *Advances in music information retrieval*, pp. 364–388, Springer, 2010.
- [186] Pearce, M. T. and Wiggins, G. A., Expectation in melody: The influence of context and learning, *Music Perception: An Interdisciplinary Journal*, 23(5):377–405, 2006.
- [187] Peretz, I. and Babaï, M., The role of contour and intervals in the recognition of melody parts: Evidence from cerebral asymmetries in musicians, *Neuropsychologia*, 30(3):277–292, 1992, ISSN 0028-3932.
- [188] Peterson, E. M., Creativity in music listening, *Arts Education Policy Review*, 107(3):15–21, 2006.
- [189] Potter, T., All my children: A portrait of Sir Andrzej panufnik based on conversations with Tully Potter, *The Musical Times*, 132(1778):186–191, 1991.
- [190] Powers, H. S., Language models and musical analysis, *Ethnomusicology*, 24(1):1–60, 1980.
- [191] Pressnitzer, D., Suied, C., and Shamma, S. A., Auditory scene analysis: the sweet music of ambiguity, *Frontiers in human neuroscience*, 5, 2011.
- [192] Purwins, H., Grachten, M., Herrera, P., Hazan, A., Marxer, R., and Serra, X., Computational models of music perception and cognition II: Domain-specific music processing, *Physics of Life Reviews*, 5(3):169–182, 2008.
- [193] Purwins, H., Herrera, P., Grachten, M., Hazan, A., Marxer, R., and Serra, X., Computational models of music perception and cognition I: The perceptual and cognitive processing chain, *Physics of Life Reviews*, 5(3):151–168, 2008.
- [194] Rabiner, L. and Juang, B., An introduction to hidden Markov models, *ASSP Magazine, IEEE*, 3(1):4–16, 1986.
- [195] Ralph, P. and Wand, Y., A proposal for a formal definition of the design concept, in *Design requirements engineering: A ten-year perspective*, pp. 103–136, Springer, 2009.
- [196] Rao, A. and Georgeff, M., BDI agents: From theory to practice, in *Proceedings of the first international conference on multi-agent systems (ICMAS-95)*, pp. 312–319, San Francisco, 1995.
- [197] Ritchie, G., Assessing creativity, *Institute for Communicating and Collaborative Systems*, 2001.

- [198] Ritchie, G., Some empirical criteria for attributing creativity to a computer program, *Minds and Machines*, 17(1):67–99, 2007.
- [199] Rohrmeier, M., A generative grammar approach to diatonic harmonic structure, in *Proceedings of the 4th Sound and Music Computing Conference*, pp. 97–100, 2007.
- [200] Rohrmeier, M., Rebuschat, P., and Cross, I., Incidental and online learning of melodic structure, *Consciousness and cognition*, 20(2):214–222, 2011.
- [201] Rubin, D. C., A basic-systems approach to autobiographical memory, *Current Directions in Psychological Science*, 14(2):79–83, 2005.
- [202] Saffran, J. R., Johnson, E. K., Aslin, R. N., and Newport, E. L., Statistical learning of tone sequences by human infants and adults, *Cognition*, 70(1):27–52, 1999.
- [203] Saupe, K., Koelsch, S., and RübSamen, R., Spatial selective attention in a complex auditory environment such as polyphonic music, *The Journal of the Acoustical Society of America*, 127:472, 2010.
- [204] Scaletti, C. and Johnson, R., An interactive environment for object-oriented music composition and sound synthesis, *ACM SIGPLAN Notices*, 23(11):222–233, 1988.
- [205] Scheirer, E. D., *Music-listening systems*, Ph.D. thesis, Massachusetts Institute of Technology, 2000.
- [206] Schellenberg, E. G., Simplifying the implication-realization model of melodic expectancy, *Music Perception*, pp. 295–318, 1997.
- [207] Schillinger, J., *The Schillinger system of musical composition*, vol. 2, Da Capo Press, 1978.
- [208] Schmidhuber, J., Driven by compression progress: A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes, in *Anticipatory Behavior in Adaptive Learning Systems*, pp. 48–76, Springer, 2009.
- [209] Schmidhuber, J., Formal theory of creativity, fun, and intrinsic motivation (1990–2010), *Autonomous Mental Development, IEEE Transactions on*, 2(3):230–247, 2010.
- [210] Schmuckler, M. A., Expectation in music: Investigation of melodic and harmonic processes, *Music Perception*, pp. 109–149, 1989.
- [211] Schulkind, M. D., POSNER, R. J., and Rubin, D. C., Musical features that facilitate melody identification: How do you know it’s “your” song when they finally play it?, *Music Perception*, 21(2):217–249, 2003.
- [212] Shannon, C. E. and Weaver, W., A mathematical theory of communication, 1948.

- [213] Simon, H. A. and Newell, A., Human problem solving: The state of the theory in 1970., *American Psychologist*, 26(2):145, 1971.
- [214] Simonton, D. K., Thematic fame, melodic originality, and musical zeitgeist: A biographical and transhistorical content analysis., *Journal of Personality and Social Psychology*, 38(6):972, 1980.
- [215] Smith, B. D. and Garnett, G. E., Improvising musical structure with hierarchical neural nets, in *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2012.
- [216] Snyder, B., *Music and memory: an introduction*, The MIT Press, 2000, ISBN 0262692376.
- [217] Sober, E., The principle of parsimony, *The British Journal for the Philosophy of Science*, 32(2):145–156, 1981.
- [218] Stevens, C., Cross-cultural studies of musical pitch and time, *Acoustical science and technology*, 25(6):433–438, 2004.
- [219] Stevens, C. J., Music perception and cognition: A review of recent cross-cultural research, *Topics in cognitive science*, 4(4):653–667, 2012.
- [220] Stravinsky, I., *Poetics of music: In the form of six lessons*, vol. 66, Harvard University Press, 1970.
- [221] Stravinsky, I., Palmer, T., Elliott, N., and Bragg, M., Once, at a border ...: Aspects of Stravinsky., Kultur International Films, 1980.
- [222] Sun, R., Accounting for the computational basis of consciousness: A connectionist approach* 1, *Consciousness and Cognition*, 8(4):529–565, 1999, ISSN 1053-8100.
- [223] Swanson, H. and Berninger, V., Working memory as a source of individual differences in children's writing, *Children's writing: Towards a process theory of development of skilled writing*, pp. 31–56, 1994.
- [224] Taube, H., Common Music: A music composition language in Common Lisp and CLOS, *Computer Music Journal*, 15(2):21–32, 1991.
- [225] Temperley, D., *The cognition of basic musical structures*, MIT press, 2004.
- [226] Temperley, D., *Music and probability*, MIT Press Cambridge, 2007.
- [227] Tenney, L., J. Polansky, Temporal gestalt perception in music, *Journal of Music Theory*, 24(2), 1980.
- [228] Thom, B., Spevak, C., and Höthker, K., Melodic segmentation: Evaluating the performance of algorithms and musical experts, in *Proceedings of the International Computer Music Conference*, pp. 65–72, Citeseer, 2002.

- [229] Trehub, S. E., Schellenberg, E. G., and Kamenetsky, S. B., Infants' and adults' perception of scale structure., *Journal of experimental psychology: Human perception and performance*, 25(4):965, 1999.
- [230] Truax, B., The PODX system: interactive compositional software for the DMX-1000, *Computer Music Journal*, pp. 29–38, 1985.
- [231] Vaggione, H., Some ontological remarks about music composition processes, *Computer Music Journal*, 25(1):54–61, 2001.
- [232] van Gigch, J. P., Explaining creativity through metacreation and paradigm displacement, in *Proceedings of the 30th Annual Meeting—Society for General Systems Research*, 1986.
- [233] Veloso, M., Carbonell, J., Perez, A., Borrajo, D., Fink, E., and Blythe, J., Integrating planning and learning: The PRODIGY architecture, *Journal of Experimental and Theoretical Artificial Intelligence*, 7:81–81, 1995, ISSN 0952-813X.
- [234] Ventrella, J., Evolving structure in Liquid Music, *The Art of Artificial Evolution*, pp. 269–288, 2008.
- [235] Vernon, D., Metta, G., and Sandini, G., A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents, *Evolutionary Computation, IEEE Transactions on*, 11(2):151–180, 2007, ISSN 1089-778X.
- [236] Wallas, G., *The art of thought.*, 1926.
- [237] Walsh, D., Occam's Razor: A principle of intellectual elegance, *American Philosophical Quarterly*, 16(3):241–244, 1979.
- [238] Wang, G., Cook, P. et al., ChuckK: A concurrent, on-the-fly audio programming language, in *Proceedings of International Computer Music Conference*, pp. 219–226, 2003.
- [239] Waschka, I., R.(2007).“composing with genetic algorithms: GenDash.”, *Evolutionary Computer Music*, pp. 117–136.
- [240] Webster, M., Merriam-webster online dictionary, 2006, URL <http://www.merriam-webster.com>.
- [241] Wertheimer, M., *Gestalt theory*, Hayes Barton Press, 1938.
- [242] Wiggins, G. A., Pearce, M. T., and Müllensiefen, D., Computational modelling of music cognition and musical creativity, *Oxford handbook of computer music*, pp. 383–420, 2009.
- [243] Wiggins, J., Compositional process in music, *International handbook of research in arts education*, pp. 453–476, 2007.

- [244] Williamson, V. J., McDonald, C., Deutsch, D., Griffiths, T. D., and Stewart, L., Faster decline of pitch memory over time in congenital amusia, *Advances in Cognitive Psychology*, 6(1):15–22, 2010.
- [245] Winkler, I., Cowan, N., Csépe, V., Czigler, I., and Näätänen, R., Interactions between transient and long-term auditory memory as reflected by the mismatch negativity, *Journal of Cognitive Neuroscience*, 8(5):403–415, 1996.
- [246] Woods, W., Transition network grammars for natural language analysis, *Communications of the ACM*, 13(10):591–606, 1970.
- [247] Wray, R. and Jones, R., An introduction to Soar as an agent architecture, *Cognition and multi-agent interaction: From cognitive modeling to social simulation*, pp. 53–78, 2005.
- [248] Wright, J. K. and Bregman, A. S., Auditory stream segregation and the control of dissonance in polyphonic music, *Contemporary Music Review*, 2(1):63–92, 1987.
- [249] Xenakis, I. and Brown, R., Concerning time, *Perspectives of New Music*, pp. 84–92, 1989.
- [250] Young, R. and Lewis, R., The Soar cognitive architecture and human working memory, *Models of working memory: Mechanisms of active maintenance and executive control*, p. 224, 1999.
- [251] Zatorre, R. J. and Halpern, A. R., Mental concerts: musical imagery and auditory cortex, *Neuron*, 47(1):9–12, 2005.
- [252] Zbikowski, L. M., Musical coherence, motive, and categorization, *Music Perception*, pp. 5–42, 1999.
- [253] Zicarelli, D., M and Jam Factory, *Computer Music Journal*, 11(4):13–29, 1987.

Appendix A

Score Example 1: *experiri*

The following score for string quartet, *experiri*, was composed in ManuScore, during a directed composition study with composer/Professor Owen Underhill, as discussed in Chapter 7 (Section 7.3). Because ManuScore does not produce printable music notation as output, the transcription used for concert performance is presented here. The version of ManuScore used for this work did not include the complete MusiCog architecture, but rather employed a simple melodic segmentation algorithm as a preprocessor for the Closure-based Cueing Model (CbCM) learning algorithm and data structure (which, in a later revision, became the LTM for MusiCog). This score demonstrates a broad conception of CAC, and should generally be considered a human-composed work, with a degree of influence from the various functions offered in ManuScore, including the generative capabilities of the CbCM (as discussed in Chapter 7).

experiri
for string quartet

♩ = 84

James Beckwith Maxwell 2011

Violin I
Violin II
Viola
Violoncello

Violin I
Violin II
Vla.
Vc.

Violin I
Violin II
Vla.
Vc.

Copyright © James B. Maxwell 2011

11 *(tr)* $\text{♩} = 164$

Vln. I *fff* *p* *pp* *mp*

Vln. II *fff* *mp* *pp* *mp* *pp*

Vla. *fff* *p* *pp* *p*

Vc. *fp ff fff* *p* *pp* *mp*

rall. 19 $\text{♩} = 82$

Vln. I *mf p* *mf* *>mp* *f* *<fff*

Vln. II *mf* *>p* *f* *>mp* *f* *<fff* *mf*

Vla. *mp* *>p* *mf* *>mp* *f* *<fff* *mp*

Vc. *mf p* *mf* *>mp* *f* *<fff*

28

Vln. I *mf* *f*

Vln. II *f*

Vla. *mf* *f*

Vc. *mf* *ff*

poco rall. $\text{♩} = 168$

31

Vln. I

Vln. II

Vla.

Vc.

ff

fff

ff

ff

sul G

36

Vln. I

Vln. II

Vla.

Vc.

f

fp

f

fp

44

Vln. I

Vln. II

Vla.

Vc.

f

fp

f

f

52

Vln. I

Vln. II

Vla.

Vc.

p

f

60

Vln. I

Vln. II

Vla.

Vc.

fp

f

68

Vln. I

Vln. II

Vla.

Vc.

mf

mf

p

f

75

Vln. I Pizz. *f* arco *f*

Vln. II Pizz. *f*

Vla. *f* *p*

Vc. *p* *f*

82

Vln. I *fp* arco *tr* *fff*

Vln. II *tr* *ff* *fff*

Vla. *f* *ff* *fff*

Vc. *fp* *mp* *f* *ff* *fff*

89

Vln. I *mp*

Vln. II *mp*

Vla. *mp*

Vc. *mp*

97

Vln. I

Vln. II

Vla.

Vc.

fp

fp

fp

fp

106

$\text{♩} = \text{♩} = 112$
senza vib.

Vln. I

Vln. II

Vla.

Vc.

mf *fp* (*pp*) *ppp*

p *fp* (*pp*) *ppp*

p *fp* (*pp*) *ppp*

p *fp* (*pp*) *ppp*

senza vib.

senza vib.

senza vib.

senza vib.

p *fp* (*pp*) *ppp* *mf* *fp* > *ppp* *mf* > *p* *ppp*

119

ord.

Vln. I

Vln. II

Vla.

Vc.

mf *fp* *ppp*

fp *ppp*

fp *ppp*

fp *ppp*

senza vib.

senza vib.

senza vib.

senza vib.

fp *ppp* *mf* *fp* *mp*

125

Vln. I

Vln. II

Vla.

Vc.

fp *ppp* *fp* *pp* *fp* *mp* *pp*

Pizz. *mp* *Pizz.* *mp* *Pizz.* *pp*

sul pont.

128

Vln. I

Vln. II

Vla.

Vc.

pp *pp* *pp* *pp*

arco *arco sul tasto* *arco sul tasto*

$\text{♩} = 60$

137

Vln. I

Vln. II

Vla.

Vc.

pppp *p* *ppp* *mp*

arco *molto espr.*

145

Vln. I

Vln. II

Vla.

Vc.

mf

f

ff

3

3

3

5

ord.

ord.

150

Vln. I

Vln. II

Vla.

Vc.

6

6

6

6

6

6

6

6

Pizz.

ff

3

3

3

3

3

3

arco

3

153

Vln. I

Vln. II

Vla.

Vc.

6

6

6

6

6

6

6

6

3

fp

3

fp

6

6

6

155

Vln. I *fp* 3 3 3 *fp* 6 6

Vln. II 6 6 6 *ff* 6 3 *Pizz. *as tight as possible

Vla.

Vc.

157

Vln. I *f* *p* 6 3 3 3

Vln. II 3

Vla.

Vc.

162

Vln. I *mp* 3 3 *pp* *ppp* rall. sul pont.

Vln. II arco *ppp* sul pont. *ppp*

Vla.

Vc.

166

Vln. I

Vln. II

Vla.

Vc.

pppp

pppp

ppp
Pizz.

sul tasto

sul tasto

Detailed description: This is a page of a musical score for a string quartet, starting at measure 166. The score is written for four instruments: Violin I (Vln. I), Violin II (Vln. II), Viola (Vla.), and Violoncello (Vc.). The time signature is 3/4. The key signature has one flat (B-flat). The Vln. I and Vln. II parts feature long, sweeping melodic lines with fermatas and are marked with *pppp* (pianississimo). Both violin parts include the instruction "sul tasto" (sul tasto) above the notes. The Vla. part has a long, sustained chord with a fermata, marked with *ppp*. The Vc. part has a pizzicato (Pizz.) section with a fermata, marked with *ppp*. The score is divided into measures by vertical bar lines.

Appendix B

Score Example 2: *factura*

The score for *factura*, for percussion solo, was composed in 2012, using the current version of ManuScore. This version included the first build of the complete MusiCog architecture, and this work therefore features more extensive use of autonomous generation from MusiCog, particularly in *factura iii*. The material generated by MusiCog was edited quite extensively during the interactive composition process, primarily to fine-tune the specific pitch content, but the contour generally remains faithful to MusiCog's output.

factura i

James B. Maxwell 2012

$\text{♩} = 90$ Rubato
4 Two-tone mallets

Marimba

6

11

15

20

24

27

31

Mar.

Mar.

Mar.

Mar.

Mar.

Mar.

Mar.

ff *5* *p* *ff* *6* *p* *3* *f* *p* *pp*

f *mp* *pp* *mf* *p* *fp* *mf* *p* *ff* *p* *f* *p* *f* *p*

pp *ff* *6* *p* *mf* *p* *f*

mp *ff* *mp* *ff* *mp* *ff* *p* *ff* *mp* *fp* *ff*

p *ff* *mp* *ff* *p* *fp* *ff* *6* *p* *f* *p* (sub)

f *p* *f* *p* *f* *fp* *f* *p* *f* *p* *f* *p*

f *p* *f* *p* *f* *p* *f* *p* *f* *p* *f*

p

pp

Copyright © James B. Maxwell 2012

35

Mar.

pp

39

Mar.

f 5 *p* *f* 6 *p*

42

Mar.

pp *ppp*

factura ii

♩ = 72

James B. Maxwell 2012

4 Two-tone mallets

Vibraphone

49

Vib.

52

Vib.

55

Vib.

58

Vib.

62

Vib.

64 Vib.

66 Vib.

68 Vib.

70 Vib.

72 Vib.

75 Vib.

81 Vib.

85 Vib.

l.v.

factura iii

James B. Maxwell 2012

♩ = 106

2 Two-tone mallets

Ankle Bells

Stomp with heel

ff *fp* *mf* *ff* *mf* *p*

mf Shape the dynamic with the Marimba part

A. B.

ff *mf* *ff* *fp* *mf* *ff* *fp* *mp* *ff* *mf*

A. B.

mp *f* *ff* *mf* *p* *ff* *fp* *mp* *ff* *p* *ff* *fp*

A. B.

mp *ff* *mp* *f* *fp* *mp* *ff* *mf* *p*

A. B.

ff *ff* *mp* *ff* *ff* *mp* *ff* *mf* *p* *ff*

A. B.

fp *p* *ff* *mf* *ff* *mp* *ff* *mp*

108

A. B.

ff *mp* *ff* *p* *f* *mp* *ff*

111

A. B.

mp *ff* *mp* *ff* *mp* *ff* *fp*

114

A. B.

ff *fp* *ff* *p* *ff* *mp* *ff* *mp*

117

A. B.

p *f* *fp* *f* *ff* *mf* *ff* *fp* *ff* *fp* *f* *fp*

120

A. B.

f *fp* *ff* *mp* *ff* *ff* *mp* *ff* *mp*

123

A. B.

ff *fp* *ff* *fp* *ff* *fp* *ff* *fp* *ff* *fp* *ff* *mf* *accel.*

126

A. B.

ff *fp* *ff* *fp* *mf* *p* *ff* *fp* *ff* *fp*

128 = 120

A. B.

ff *fp* *mp* *ff* *fp* *mf* *ff* *fp*

130

A. B.

ff *fp* *ff* *fp* *ff* *fp* *mp* *ff* *fp*

132

A. B.

mp *ff* *mf* *fp* *f* *ffp* *f* *ffp* *ff* *fp*

134

A. B.

ff *mf* *ffp* *ff* *ffp* *ff* *fff*

136

A. B.

fff