

MUSICAT: A COMPUTER MODEL OF
MUSICAL LISTENING AND ANALOGY-MAKING

Eric Paul Nichols

Submitted to the faculty of the University Graduate School
in partial fulfillment of the requirements
for the degree
Doctor of Philosophy
in the Departments of Computer Science and Cognitive Science,
Indiana University
December 2012

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree Doctor of Philosophy.

Doctoral Committee:

Dr. Douglas R. Hofstadter
(Principal Adviser)

Dr. Eric Isaacson

Dr. Donald Byrd

Dr. Michael Gasser

December 7, 2012

Copyright © 2012

Eric Paul Nichols

For Helga and Steve

Acknowledgements

So many people have been friends and colleagues, providing emotional and academic support over my eight years in Bloomington, it seems difficult to list them all. Please accept my apologies for any omissions.

First, I would like to thank my immediate family for a lifetime of support: my parents Ann and Clay and my three little sisters Jenni, Sara, and Julie. I'm lucky to have fairly frequent communication with everyone thanks to email and cell phones; in a way I feel closer to my family in Bloomington, thanks to technology, than when I was an undergraduate at Montana State in Bozeman. A side note: my mom used to run a cat-breeding business, and we gave musical names to all the cats. The business was named "Mewsicats Cattery", an amusing source of inspiration for the name of my project (Musicat).

Second, I owe a debt of gratitude to a host of my professors at Indiana University. My first class in Bloomington was an excellent artificial intelligence course taught by David Leake that got me off to a great start. Soon after I started at IU, Chris Raphael joined the faculty, and after auditing some of his courses, he and I had some very successful collaborations on various music informatics projects. I consider him an unofficial member of my thesis committee, and I'm happy about the hours of mathematical and musical discussions we've had over the years. My official committee has also been extremely helpful. Eric Isaacson taught a seminar on Music Cognition during my first semester, which, in combination with Dr. Leake's course, was a perfect way to start my grad school career; it gave me a great introduction to the field. Mike Gasser's courses on biologically-inspired computing and natural language processing were also extremely insightful. Don Byrd was a great musical presence in Music Informatics group meetings and at ISMIR conferences, and always has given me great feedback on this project. Finally, it goes without saying that my

main advisor, Doug Hofstadter, has been a tremendous influence — not only during my eight years at IU, but indeed ever since I was 18 and started reading *Gödel, Escher, Bach*. I'm especially grateful that he made it possible for me to tag along to Paris in 2010 — *il m'a emmené dans ses valises*, as they say — when he spent a semester there on sabbatical. I expected to learn about artificial intelligence when I came to Bloomington, but Doug actually taught me about cognitive science, communication, writing, and even French pronunciation, among many other things. It's truly been an honor and a privilege!

Third, thanks to my many grad student compatriots from Bloomington: current and past students including Mike Brady, Trista Chan, Annet Czeplédi, Yupeng Gu, Yushen Han, Chris Harshaw, Chris Honey, Andrew Kalafut, Ian Knopke, Olga Rass, and Toshi Uchino; fellow FARGonauts Dave Bender, Harry Foundalis, Ben Kovitz, Francisco Lara-Dammer, Abhijit Mahabal, Damien Sullivan, and Will York; and cyclists extraordinaires George Kachergis and Richard Veale. While in Paris, although I was only there for a semester, I was fortunate to make four great friends: Cinthia Altieri, Tatiana Catanzaro, Rim Roudies, and Junie Terrier. Special thanks to my FARGonaut friends and housemates Matthew Hurley and Alexandre Linhares, both of whom were extremely influential in both my academic and personal life — many great discussions, fun times, crazy projects, and general hilarity!

Fourth, I would like to thank my other close friends in Bloomington (and elsewhere), who formed a sort of surrogate extended family. Alia Alkasimi quickly became my best friend and we've remained like family even after she moved out of town — *merci pour tous, liwliw*. She has also been one of the best sources of moral support helping me succeed with this dissertation, not to mention life in general. Other dear friends include Annika Wallendahl, Lisa Cooper, Zach Saul, Jing Pan, Claire Houck and her dog Jonie, the musically gifted, wonderful Welsh girl Sara Angharad, the incomparable human Lindsay “Z” Arcurio, my “alien twin” Quynh Neutron, my extreme-waltz partner Katherine Wiley, and, of course, the

amazing and artistic musician Priscilla Borges — *obrigado por todos as cores!* Thanks to you all for all the support, friendship, and love.

Finally, on a more somber note, thanks are due to both Helga Keller and Steve Larson, but unfortunately they both passed away in 2011 — a double blow to the CRCC family. Every dissertation like this one from FARG mentions Helga's support and contribution to the Center and her great impact on the lives of the graduate students at FARG. Helga was instrumental in my decision to come to Indiana University, not only by putting me in touch with Doug Hofstadter via phone but also by sending me a surprisingly large collection of CRCC technical reports, which made me very excited to move to Bloomington. Throughout my Ph.D., she was a tremendous help with administrative matters, including lots of extra work on an NSF grant application, but more importantly I enjoyed dinner parties she hosted, a campus tour she gave to my parents, random conversations on quiet days at the office, and a spontaneous road trip to southern Indiana with her and Howard. Helga's presence at CRCC, and in Bloomington, is deeply missed.

Steve Larson, like Helga, directly influenced my decision to come to IU. We met online when he responded to a discussion group post in which I asked where I should apply to grad school. He recommended that I read *Fluid Concepts and Creative Analogies* and encouraged me to apply to IU in Bloomington. We met in person in Eugene, Oregon and discussed his Seek Well project, which led to this dissertation. My meetings in person and via phone with Steve over the years were extremely valuable and his ideas will continue to influence those of us in music cognition for years to come. Thanks, and so long, Steve.

In addition to all the social support mentioned above, I would also like to acknowledge several sources of financial support for this work. I was generously supported by fellowships from CRCC and from the Music Informatics program at IU, as well as by NSF Grant IIS-0738384, awarded by the NSF's CreativeIT program.

Musicat: A Computer Model of Musical Listening and Analogy-Making

What happens when people listen to music? What sorts of mental structures are formed? How do we make sense of a melody as its notes fly by in rapid succession? Can we model the experience of listening to music in real time?

This dissertation describes the computer program Musicat, which was designed to explore these questions. Musicat is a model of real-time melody perception by people. The program “listens” to monophonic Western tonal melodies one note at a time (presented not as audio recordings, but rather in a symbolic form much like sheet music) and generates an internal representation of the musical structures it “hears”. These structures include groups of adjacent notes, meta-groups comprised of smaller groups, expectations for upcoming structures, and, most importantly, analogies between groups (and meta-groups) of various sizes. In the model, listening is not a passive process; instead, it is an active, dynamic process of creating mental structures. Thus when Musicat listens to a melody, I consider such an act to be creative, and I call it a “listening performance”.

Musicat follows in the tradition of previous computer models of the Fluid Analogies Research Group: it is based on the architectures of Copycat, Tabletop, and related programs, with modifications made to accommodate music’s temporal nature. In addition to giving a technical discussion of Musicat’s architecture, this dissertation includes copious examples of the program’s listening performances on a variety of melodies ranging from simple children’s songs to more complex popular-song forms. A discussion of the results suggests that Musicat captures several previously unmodeled aspects of musical listening, such as analogy-making and hierarchical grouping in real time, but that much more work is needed to achieve humanlike listening performances on even the simplest of melodies.

Table of Contents

Acknowledgements.....	ix
Table of Contents.....	xv
List of Figures.....	xxiii
List of Tables	xxxi
CHAPTER ONE	1
INTRODUCTION	1
Whence Musicat?	1
What is Musicat?	2
Wherefore Musicat?.....	4
Music-listening as Performance	5
Music-listening is Dynamic.....	5
Music-listening is a Creative Process.....	8
Music-listening is Not a Universal Talent	12
Overview of this Dissertation	14
Whither Musicat?.....	15
CHAPTER TWO	17
WORK THAT HAS INFLUENCED THIS PROJECT	17
Models of Melodic Expectation and Cognition	17
Leonard Meyer.....	17
Eugene Narmour.....	22
Adam Ockelford.....	30

Olivier Lartillot	31
Fred Lerdahl and Ray Jackendoff	33
Fred Lerdahl	37
Diana Deutsch and John Feroe	39
David Temperley	40
Elizabeth Margulis	42
David Huron	46
Bob Snyder	50
Steve Larson	58
Musical Rhythm	67
Dirk-Jan Povel and Peter Essens	68
Emilios Cambouropoulos	69
William Rothstein	70
FARG	75
Copycat (Melanie Mitchell)	76
Metacat (James Marshall)	81
Numbo (Daniel Defays)	81
Tabletop (Robert French)	83
Letter Spirit (Gary McGraw and John Rehling)	86
Phaeaco (Harry Foundalis)	87
Seek-Whence (Marsha Meredith) and Seqsee (Abhijit Mahabal)	87
Seek Well (Steve Larson)	89
Capyblanca (Alexandre Linhares)	91

CHAPTER THREE	95
MUSICAT’S DOMAIN	95
Introduction.....	95
A Note on the Universality of Musicat’s Domain.....	96
Microdomain Specifics	97
Melody Characteristics	97
Extra Information Included with Melodies.....	98
Two Examples	98
Example Melodies and Human Listening Performances	100
Mystery Melody	100
Sur le pont d’Avignon	108
On the Street Where You Live	115
A Sneak Preview of Musicat.....	124
CHAPTER FOUR	125
MUSIC AND ANALOGY	125
Analogies All the Way Down.....	125
Ascending Through the Levels.....	126
Sound Waves.....	127
Timbre	129
Pitch	130
Rhythm.....	132
Motif.....	134
Phrases	138
Large-scale Form	139
Raindrops Keep Falling... ..	140

Analogy as the Core of Music Cognition	142
Analogy as the Core of Musicat	143
The Importance of Analogy in Models of Music Cognition	143
CHAPTER FIVE	147
MUSICAT LISTENS TO BAD MELODIES	147
Understanding Musicat's Listening Performances	149
Some Simplified Notation	153
A Note on the Pronouns "I" and "We"	154
First Example: Twinkle, Twinkle, Little Star	155
Bad Melodies	173
Bad Melody #1	174
Bad Melody #2	179
Bad Melody #3	184
Bad Melody #4	187
Bad Melody #5	193
Summary	195
CHAPTER SIX	197
MUSICAT LISTENS TO SIMPLE MELODIES	197
Twinkle, Twinkle, Little Star (run 2)	198
Row, Row, Row Your Boat	208
Sur le pont d'Avignon	221
Frère Jacques	230
Sicilienne (Fauré)	237

CHAPTER SEVEN	247
MUSICAT LISTENS TO COMPLEX MELODIES	247
Younger than Springtime (Rodgers and Hammerstein).....	247
On the Street Where You Live (Lerner and Loewe).....	263
Tennessee Waltz (Stewart and King)	284
Good People All (Hofstadter)	303
Sun and Moon (Boublil and Schönberg)	312
CHAPTER EIGHT	331
THE ARCHITECTURE OF MUSICAT	331
Overview: Mental Processes to Model	332
Major Components.....	334
User Interface.....	334
Program Initialization and Assumptions	338
Main Program Loop.....	340
Workspace.....	342
Coderack.....	343
Codelets	346
Temperature.....	351
Sliding Time Window of Perception.....	353
Objects in the Workspace.....	355
Notes	355
Measures	356
Rhythm-Based Measure Links	356
Relationships.....	357
Bar Lines	358
Alphabets	359

Groups and Meta-groups.....	359
Analogies.....	363
Expectations.....	365
Calculating Structure Strengths	366
Scoring Rhythmic Measure Links.....	369
Scoring Relationships	370
Scoring Groups	374
Scoring Sequences	378
Scoring Analogies	379
Scoring Expectations	381
Creating Structures with Codelets	382
Rhythm-based Measure Links	382
Relationships.....	383
Bar Lines	385
Groups and Meta-groups.....	386
Analogies.....	388
Expectations.....	391
CHAPTER NINE	393
THE EVOLUTION OF MUSICAT	393
A Brief History of Musicat	394
Versions of Musicat	396
Two-note Model (Key and Meter Induction).....	396
Musicat with Hierarchy (MusicatH).....	397
Musicat Without Hierarchy (MusicatW).....	416
BinaryCat.....	445
RhythmCat	447

CHAPTER TEN	455
CONCLUSIONS AND FUTURE WORK	455
Aims of the project.....	455
What Musicat Does Well.....	456
Musicat Behaves Differently on “Bad” Melodies.....	456
Musicat Generates Hierarchical Grouping Structures	457
Musicat Listens in Real Time with Limited Working Memory	458
Musicat Notices Different Things on Different Runs.....	460
Musicat Forms Analogies.....	461
What Musicat Does Badly.....	462
Musicat Forms Groups at Measure Boundaries Only.....	463
Musicat Misses Too Many Simple Things	463
Musicat Makes Errors in Grouping and Bar-line Thicknesses	464
Musicat Makes Too Many Analogies	465
Musicat Makes Too Few Analogies	466
Musicat Ignores Many Crucial Aspects of Pitch.....	466
Musicat Doesn’t Generate Note-level Expectations	467
Musicat Flails	468
Musicat Destroys Large Structures.....	470
Musicat Gets Confused by Long Melodies	471
Contributions of Musicat	472
Lessons Learned	472
Architectural Contributions	475
Contributions to the Modeling of Music Cognition.....	477
Future Work.....	482
Next Steps	482

A Hypothesis for Future Testing.....	483
Whither Musicat? (Questions and Speculations about the Future).....	485
Could Musicat be Extended to Listen to Polyphonic Music?	485
Will Musicat Ever Listen to Audio Recordings?	485
Will Musicat Ever Hear Music in the Same Way as a Human Does?.....	486
Will Musicat Ever Become a Composer?	486
REFERENCES	489
PILOT STUDY	493
PRELIMINARY QUANTITATIVE RESULTS	501

List of Figures

Figure 1.1: Opening motif of Beethoven's Fifth Symphony.	2
Figure 1.2: Opening motif, including the next four notes.	3
Figure 1.3: Previous figure with groups and an analogy.	4
Figure 1.4: Opening of “Ants Marching”.	6
Figure 1.5: Opening melodic motif in “Ants Marching”.	6
Figure 1.6: Rhythmic reinterpretation of Figure 1.5: drum on beats 2 and 4, not 1 and 3. ...	7
Figure 2.1: Melody described using pitch alphabets.	39
Figure 2.2: Auditory memory processes, adapted from (Snyder 2000).	51
Figure 2.3: Beginnings of typical Seek Well melodies.	59
Figure 2.4: Three musical forces.	60
Figure 2.5: Ascending octave.	60
Figure 2.6: Alternate melodic continuations after C-D.	63
Figure 2.7: A simple melody (below) and an implied background level (above).	65
Figure 2.8: Prediction at a higher level.	66
Figure 2.9: Surface-level prediction.	66
Figure 2.10: No boundaries.	69
Figure 2.11: Two boundaries.	69
Figure 3.1: “Sur le pont d’Avignon”, a typical melody in Musicat’s domain.	99
Figure 3.2: “On the Street Where You Live”, a more complicated melody in the domain. ..	99
Figure 3.3: Mystery Melody, Note 1.	101
Figure 3.4: Mystery Melody, Notes 1–2.	101
Figure 3.5: Mystery Melody, Notes 1–3.	102
Figure 3.6: Mystery Melody, Notes 1–4.	103
Figure 3.7: Mystery Melody, Notes 1–5.	103
Figure 3.8: Mystery Melody, Notes 1–6.	104
Figure 3.9: Mystery Melody, Notes 1–5, heard in 4/4.	105
Figure 3.10: Mystery Melody, Notes 1–5, heard in 3/4.	105
Figure 3.11: Mystery Melody, Notes 1–7.	106
Figure 3.12: Mystery Melody, Notes 1–8.	106
Figure 3.13: Mystery Melody, Notes 1–9.	106
Figure 3.14: Mystery Melody, Notes 1–9, heard in 3/4.	106
Figure 3.15: An unrealistic expectation for the next three notes.	107
Figure 3.16: Mystery Melody, Notes 1–10 (complete).	107
Figure 3.17: Opening theme of Bach’s Organ Prelude in C major, BWV 547.	108
Figure 3.18: “Sur le pont d’Avignon”, measure 1.	109

Figure 3.19: “Sur le pont d’Avignon”, measures 1–2.	109
Figure 3.20: “Sur le pont d’Avignon”, measures 1–3.	110
Figure 3.21: “Sur le pont d’Avignon”, measures 1–4.	111
Figure 3.22: “Sur le pont d’Avignon”, measures 5–7.	113
Figure 3.23: “Sur le pont d’Avignon”, complete.	114
Figure 3.24: “On the Street Where You Live”, first 5 notes.	115
Figure 3.25: “On the Street Where You Live”, first phrase.	116
Figure 3.26: “On the Street Where You Live”, second phrase.	118
Figure 3.27: “On the Street Where You Live”, third phrase.	121
Figure 3.28: “On the Street Where You Live”, fourth phrase.	122
Figure 3.29: “On the Street Where You Live”, phrases 1–4.	123
Figure 3.30: One of Musicat’s listening performances for “Sur le pont d’Avignon”.	124
Figure 4.1: A pure tone at 440 Hz.	127
Figure 4.2: A composition of two sine waves at 440 and 880 Hz.	128
Figure 4.3: A small portion of the waveform of the note A played on a piano.	129
Figure 4.4: The beginning of Chopin’s “Raindrop” prelude.	130
Figure 4.5: The key change to C# Minor between measures 28 and 29.	131
Figure 4.6: G# becomes B in measure 41.	132
Figure 4.7: Measure 23 (analogous to measure 4).	133
Figure 4.8: Measure 85 (analogous to measure 23).	134
Figure 4.9: Ab becomes F.	135
Figure 4.10: Another variant of the repetition motif.	136
Figure 4.11: Rhythmic motif.	136
Figure 4.12: Rhythmic motif appearing in another form.	136
Figure 4.13: Another appearance of the motif.	137
Figure 4.14: Measures 8-11: High-descending-notes motif.	137
Figure 4.15: Return of the high descent motif: a ray of sunlight or “a star in the jaws of the clouds”.	138
Figure 5.1: A sample screenshot demonstrating Musicat’s notation.	150
Figure 5.2: Twinkle, Twinkle, Little Star.	155
Figure 5.3: Twinkle, Twinkle, with lyrics.	156
Figure 5.4: Twinkle, Twinkle, measure 1.	157
Figure 5.5: Twinkle, Twinkle, measure 2.	158
Figure 5.6: Twinkle, Twinkle, measure 3.	159
Figure 5.7: Twinkle, Twinkle, measure 4.	160
Figure 5.8: Twinkle, Twinkle, measure 5.	161
Figure 5.9: Twinkle, Twinkle, measure 6.	161
Figure 5.10: Twinkle, Twinkle, measure 7.	162

Figure 5.11: Twinkle, Twinkle, measure 8.....	163
Figure 5.12: Twinkle, Twinkle, measure 9.....	164
Figure 5.13: Twinkle, Twinkle, measure 10.....	165
Figure 5.14: Twinkle, Twinkle, measure 11.....	166
Figure 5.15: Twinkle, Twinkle, measure 12 (end of song).	167
Figure 5.16: Twinkle, Twinkle, processing.	169
Figure 5.17: Twinkle, Twinkle, processing.	170
Figure 5.18: Twinkle, Twinkle, final state, low detail level (strongest structures only).....	171
Figure 5.19: Another run on "Twinkle, Twinkle".....	173
Figure 5.20: Bad Melody #1 (random rhythms and notes).	174
Figure 5.21: Bad Melody #1, measure 4.	175
Figure 5.22: Bad Melody #1, measure 5.	175
Figure 5.23: Bad Melody #1, end of processing.	176
Figure 5.24: Bad Melody #1, low detail level.	177
Figure 5.25: Bad Melody #1, medium level of detail.	178
Figure 5.26: Bad Melody #2 (constrained rhythms).....	179
Figure 5.27: Bad Melody #2 (run 1), end of processing.	181
Figure 5.28: Bad Melody #2 (run 1), low level of detail.....	182
Figure 5.29: Bad Melody #2 (run 2), low level of detail.....	182
Figure 5.30: Bad Melody #2 (run 3), low level of detail.....	183
Figure 5.31: Bad Melody #3 (constrained rhythm).....	184
Figure 5.32: Bad Melody #3 (run 1).....	185
Figure 5.33: Bad Melody #3 (run 1), low detail.....	186
Figure 5.34: Bad Melody 3 (run 2), low detail.....	186
Figure 5.35: Bad Melody #4 (odd-length phrases).	187
Figure 5.36: Bad Melody #4, measure 5.	188
Figure 5.37: Bad Melody #4, measure 6.	189
Figure 5.38: Bad Melody #4, measure 7.	190
Figure 5.39: Bad Melody #4, end of processing.	191
Figure 5.40: Bad Melody #5.....	193
Figure 5.41: Bad Melody #5 (run 1).....	194
Figure 5.42: Bad Melody #5 (run 2).....	195
Figure 6.1: Twinkle Twinkle (run 2), measure 4.....	198
Figure 6.2: Twinkle, Twinkle (run 2), measure 9.....	199
Figure 6.3: Twinkle, Twinkle (run 2), measure 12.....	201
Figure 6.4: Twinkle, Twinkle (run 2), measure 12 (+3 measures of extra time).....	202
Figure 6.5: Twinkle, Twinkle (run 2), measure 12(+4).	203
Figure 6.6: Twinkle, Twinkle (run 2), end of run.	204

Figure 6.7: Twinkle, Twinkle (run 2), outer group (<i>very</i> low level of detail).....	205
Figure 6.8: Twinkle, Twinkle (run 2), big analogy (low level of detail).	205
Figure 6.9: Twinkle, Twinkle (run 2), medium detail level.	206
Figure 6.10: Twinkle, Twinkle (run 2), medium detail level. Analogy in blue singled-out.	207
Figure 6.11: Row, Row, Row Your Boat.	208
Figure 6.12: Row, Row, Row Your Boat (run 1).....	209
Figure 6.13: Row, Row, Row Your Boat (run 2).....	211
Figure 6.14: Row, Row, Row Your Boat (run 3).....	213
Figure 6.15: Row, Row, Row Your Boat (run 4).....	215
Figure 6.16: Row, Row, Row Your Boat (run 5).....	218
Figure 6.17: Row, Row, Row Your Boat (run 5), low detail level.	219
Figure 6.18: Row, Row, Row Your Boat (run 5), medium detail level.	220
Figure 6.19: Sur le pont d’Avignon.....	221
Figure 6.20: Sur le pont d’Avignon (run 1), just after hearing the final measure.	222
Figure 6.21: Sur le pont d’Avignon (run 1), low detail level.....	223
Figure 6.22: Sur le pont d’Avignon (run 1), medium detail level.	223
Figure 6.23: Sur le pont d’Avignon (run 1), high detail level.	225
Figure 6.24: Sur le pont d’Avignon (run 2), just after hearing the final measure.	226
Figure 6.25: Sur le pont d’Avignon (run 2), final state.	227
Figure 6.26: Sur le pont d’Avignon (run 2), low detail level.....	228
Figure 6.27: Sur le pont d’Avignon (run 2), medium detail level.	228
Figure 6.28: Frère Jacques.....	230
Figure 6.29: Frère Jacques d’Avignon.....	230
Figure 6.30: Frère Jacques (run 1).....	231
Figure 6.31: Frère Jacques (run 2).....	232
Figure 6.32: Frère Jacques (run 2), strongest structures.	233
Figure 6.33: Frère Jacques (run 3).....	234
Figure 6.34: Frère Jacques (run 3), low detail.....	235
Figure 6.35: Frère Jacques (run 3), medium detail.	236
Figure 6.36: Fauré’s Sicilienne for cello and piano, op. 78.....	237
Figure 6.37: Sicilienne (run 1).....	238
Figure 6.38: Sicilienne (run 2), final state.	240
Figure 6.39: Sicilienne (run 2), measure 8(+1).....	241
Figure 6.40: Sicilienne (run 2), measure 8(+2).....	241
Figure 6.41: Sicilienne (run 2), measure 8(+5).....	242
Figure 6.42: Sicilienne (run 2), final state, medium detail.....	242
Figure 6.43: Sicilienne (run 3).....	244

Figure 7.1: Younger than Springtime (from the musical <i>South Pacific</i>), 32-measure excerpt.	247
Figure 7.2: Younger than Springtime, 8 measures.	249
Figure 7.3: Younger than Springtime, 8 measures.	250
Figure 7.4: Younger than Springtime (16-measure excerpt)	251
Figure 7.5: Younger than Springtime (16 measures, run 1), after measure 9	252
Figure 7.6: Younger than Springtime (16 measures, run 1) after measure 15	253
Figure 7.7: Younger than Springtime (16 measures, run 1), end of processing.	255
Figure 7.8: Younger than Springtime (16 measures, run 2).	257
Figure 7.9: Younger than Springtime (16 measures, run 3).	258
Figure 7.10: Younger than Springtime, 8-measure bridge.	259
Figure 7.11: Younger than Springtime (middle 8 measures)	260
Figure 7.12: Younger than Springtime (bridge, run 2).	262
Figure 7.13: On the Street Where You Live (from the musical <i>My Fair Lady</i>).	263
Figure 7.14: Metrically-shifted version of “On the Street Where You Live”, illustrating the measure-numbering scheme used in the text	264
Figure 7.15: On the Street Where You Live (run 1).	266
Figure 7.16: On the Street Where You Live (run 2).	269
Figure 7.17: On the Street Where You Live (run 2), low detail.	270
Figure 7.18: On the Street Where You Live (run 3).	274
Figure 7.19: On the Street Where You Live (run 3), low detail.	275
Figure 7.20: On the Street Where You Live (run 3), medium detail	276
Figure 7.21: On the Street Where You Live (run 4).	277
Figure 7.22: On the Street Where You Live (run 5).	278
Figure 7.23: On the Street Where You Live (run 5), low detail.	279
Figure 7.24: On the Street Where You Live (run 5), medium detail	280
Figure 7.25: On the Street Where You Live (run 6), with a very large analogy!	281
Figure 7.26: Tennessee Waltz.	284
Figure 7.27: Tennessee Waltz.	285
Figure 7.28: Tennessee Waltz, first half.	286
Figure 7.29: Tennessee Waltz, first half.	287
Figure 7.30: Tennessee Waltz, first half (run 2).	289
Figure 7.31: Tennessee Waltz, first half (run 3).	291
Figure 7.32: Tennessee Waltz, second half	292
Figure 7.33: Tennessee Waltz, second half (run 1).	293
Figure 7.34: Measures 5–8 (top staff) and 13–16 (bottom staff).	295
Figure 7.35: Tennessee Waltz, second half (run 2), after measure 12.	296
Figure 7.36: Tennessee Waltz, second half (run 2), after measure 13.	297

Figure 7.37: Tennessee Waltz, second half (run 2), after measure 14.	298
Figure 7.38: Tennessee Waltz, second half (run 2), after measure 15.	299
Figure 7.39: Tennessee Waltz, second half (run 2), after measure 16(+3).	300
Figure 7.40: Tennessee Waltz, second half (run 2), end of processing.	301
Figure 7.41: Opening measures of Good People All, from a cantata by Douglas Hofstadter.	303
Figure 7.42: Good People All (measures 1–4): starting points of groups indicated by upper voice.	304
Figure 7.43: Good People All (measures 1–4): groups represented by their starting notes.	304
Figure 7.44: Good People All, recomposed.	306
Figure 7.45: Good People All (run 1).	307
Figure 7.46: Good People All (run 2).	309
Figure 7.47: Good People All (run 3).	310
Figure 7.48: Sun and Moon (from the musical <i>Miss Saigon</i>).	312
Figure 7.49: <i>Miss Saigon</i> , opening notes.	313
Figure 7.50: <i>Miss Saigon</i> , opening accented notes, rewritten at double tempo.	314
Figure 7.51: Sun and Moon, with phrasing implied by the text.	315
Figure 7.52: Sun and Moon, with an alternate possible grouping suggested by the chords.	315
Figure 7.53: Sun and Moon, first half.	316
Figure 7.54: Sun and Moon, first half (run 1).	317
Figure 7.55: Sun and Moon, first half (run 1), after measure 6.	318
Figure 7.56: Sun and Moon, first half (run 1), after measure 10.	319
Figure 7.57: Sun and Moon, measures 1-12. Descending scale shown with large noteheads.	320
Figure 7.58: Sun and Moon, first half (run 1), medium detail.	321
Figure 7.59: Sun and Moon, first half (run 2) (final processing).	322
Figure 7.60: Sun and Moon, first half (run 2), medium detail.	323
Figure 7.61: Sun and Moon, second half (measures 13–25, renumbered as 1–13 for simplicity).	324
Figure 7.62: Sun and Moon, second half (run 1).	325
Figure 7.63: Sun and Moon, second half (run 1), after measure 11.	327
Figure 7.64: Sun and Moon, second half (run 2).	328
Figure 7.65: Sun and Moon, second half (run 3).	330
Figure 8.1: Standard user interface for Musicat.	335
Figure 8.2: Development interface.	337
Figure 8.3: Hypermetric upbeat in Mozart’s 40th Symphony, First Movement.	339
Figure 8.4: Tennessee Waltz melody with pickup beat.	340
Figure 8.5: Tennessee Waltz melody with shifted bar lines.	340

Figure 8.6: Codelet-creation graph. Boxes at the top of the graph represent codelets that can generate codelets of the types below, shown by arrows.	345
Figure 8.7: A typical sequence of bar line “thicknesses”	358
Figure 8.8: A typical sequence.	362
Figure 8.9: White (Deep Blue) to move, just before move 37 (Be4). Game 2, Kasparov versus Deep Blue, 1997.	368
Figure 8.10: Twinkle, Twinkle.	371
Figure 8.11: Measures 1–2.	372
Figure 8.12: Measures 3–4.	372
Figure 8.13: Sigmoid function.	378
Figure 9.1: Opening of “Stückchen”, from Schumann’s <i>Kinderszenen</i> , Op. 15.	398
Figure 9.2: MusicatH listening to a modified version of “Stückchen”.	399
Figure 9.3: MusicatH’s Slipnet.	402
Figure 9.4: Stückchen, run 1.	404
Figure 9.5: Stückchen, run 1.	404
Figure 9.6: Stückchen, run 2.	405
Figure 9.7: Triplet Scale.	406
Figure 9.8: Triplet Scale, heard in 3/4 meter.	406
Figure 9.9: Triplet Scale, run 1.	406
Figure 9.10: Triplet Scale, run 1.	407
Figure 9.11: Triplet Scale, run 1 (end of run).	407
Figure 9.12: Triplet Scale, run 2.	408
Figure 9.13: Triplet Scale, run 3.	409
Figure 9.14: Triplet Scale, run 4.	409
Figure 9.15: Twinkle, Twinkle, run 1.	410
Figure 9.16: Twinkle, Twinkle, run 2.	411
Figure 9.17: Twinkle, Twinkle, run 2, a moment later.	411
Figure 9.18: Good People All (first two measures).	412
Figure 9.19: On the Street Where You Live, run 1.	413
Figure 9.20: On the Street Where You Live, run 2.	414
Figure 9.21: MusicatW running on “Stückchen”.	417
Figure 9.22: MusicatW running on “Stückchen”, note links hidden.	418
Figure 9.23: Global temperature.	422
Figure 9.24: Codelet urgencies during a run.	423
Figure 9.25: Real-time codelet adjustment window.	424
Figure 9.26: Bar lines of “thicknesses” following a regular pattern.	429
Figure 9.27: Analogy maps in MusicatW.	430
Figure 9.28: “Stückchen”	432

Figure 9.29: Triplet Scale.....	433
Figure 9.30: “Twinkle, Twinkle” (run 1).....	434
Figure 9.31: “Twinkle, Twinkle” (run 2).....	435
Figure 9.32: “Twinkle, Twinkle” (run 3).....	435
Figure 9.33: “On the Street Where You Live” (run 1).....	436
Figure 9.34: “On the Street Where You Live” (run 2, with analogies), mid-run.	437
Figure 9.35: “On the Street Where You Live” (run 2, with analogies), end of run.....	438
Figure 9.36: A very simple melody.....	440
Figure 9.37: Very simple melody, during final processing.....	441
Figure 9.38: Very simple melody, end of run.	442
Figure 9.39: Sur le pont d’Avignon.....	443
Figure 9.40: Sur le pont d’Avignon, result of a run using the latest Musicat (repeated from Chapter 6).....	444
Figure 9.41: Far too many analogies in “Sur le pont d’Avignon” — an embarrassment of riches.....	444
Figure 9.42: “Sur le pont d’Avignon”.....	446
Figure 9.43: “Sur le pont d’Avignon”, rhythm only.....	448
Figure 9.44: A run of an early version of RhythmCat on a simple rhythm.	449
Figure 9.45: A run of a later version of RhythmCat, whose display resembles that of the latest version of Musicat.	450
Figure 10.1: Frequency counts of tonal functions for first notes.....	497
Figure 10.2: Krumhansl and Kessler’s key profiles.....	497
Figure 10.3: Comparison of first note data with weighted Krumhansl and Kessler profiles.	498
Figure 10.4: More comparisons.	499

List of Tables

Table 1: Similarities between the two halves of the first phrase of “On the Street Where You Live”.	117
Table 2: Contour symbols.	371
Table 3: Preliminary results for Musicat, compared with reported results from CBMS.	503

CHAPTER ONE

Introduction

Whence Musicat?

In 2004, I came to graduate school at Indiana University in Bloomington with a very specific research goal: I wanted to make a computer program that could write music. More specifically, I intended to write a program that could compose its own beautiful and original melodies¹ — not algorithmic music in the tradition of Lejaren Hiller and Leonard Isaacson’s *Illiac Suite*, or “recombinant” music such as the Bach-derived inventions and Joplin-derived rags of David Cope’s EMI. I had a specific plan: I would create a composing program modeled after ideas of the program *Letter Spirit*, from the Fluid Analogies Research Group (FARG), which was intended to carry out the creative task of generating stylistically self-consistent alphabets. Specifically, I was excited by a list of bullet points on page 411 in the book *Fluid Concepts and Creative Analogies* (Hofstadter & FARG, 1995) that gives a set of requirements that must be met for a program to be called “creative”. However, after arriving in Bloomington, I quickly realized that it was premature to attempt the computer music-composition endeavor, because of a problem that became more apparent to me as a student

¹ This imaginary melody-creation machine, like George Orwell’s melody-composing “Versificator” in the novel *1984*, is just the sort of thing that Douglas Hofstadter, my advisor, doesn’t like the idea of, and doesn’t expect will happen any time soon. See (Cope & Hofstadter, 2001) for a lively discussion.

of cognitive science than it had been earlier when I was simply excited about the prospect of computers generating music: *computer programs can't listen*. My imagined computer composer would theoretically come into existence with the ability to *create* melodies yet without any ability to *hear* them. This made no sense, since I couldn't expect high-quality musical results to issue from a deaf computer. (Beethoven composed music while deaf, but he had had the experience of many decades of listening to and composing music while he could still hear.) Therefore, I refocused on an activity that is fundamental to the task of music composition: *music listening*.

What is Musicat?

This thesis thus describes my first efforts at computationally modeling human music-listening, implemented in my computer program called “Musicat”. (The name “Musicat” was inspired by the program from which the core architecture of Musicat is derived, Copycat, along with Copycat’s successors, Metacat and Magnificat.) The program simulates the process of listening to a simple melody in the Western tonal tradition, and displays on the screen the various internal cognitive structures that form as the melody progresses in time. These structures take the form of *groups of notes* and *analogies between group structures*. For example, Figure 1 shows the so-called “Fate” motif at the start of Beethoven’s Fifth Symphony. What mental representations are formed when a *person* hears these notes?



Figure 1.1: Opening motif of Beethoven's Fifth Symphony.

Several things might happen for a listener. First, after the first note is repeated, and then repeated again, any listener will notice that there are three instances of the same note in a row; these three notes might be mentally grouped together. Then, after the fourth note sounds and is held for a while, all four notes will likely be grouped together. This kind of grouping is predicted by theories of gestalt perception. Much more cognitive activity is possible, of course. The listener will notice the different fourth pitch, and will probably recognize it as downwards motion, or even motion by a certain distance (a leap of a “third”). A listener familiar with British war announcements broadcast via radio in World War II might feel a variety of complex emotions if reminded of the use of these four notes in that particular context.



Figure 1.2: Opening motif, including the next four notes.

After the fermata, things get a bit more interesting (Figure 1.2). Several of the same processes occur when a person hears these next four notes: the first three notes may be heard as being three instances of the same pitch, they may be grouped together, the following note will sound like a leap downwards, and so on. But the pitches are different; this new passage sounds like a version of the first passage that has been shifted down in pitch. After the music reaches the second fermata, the listener clearly will have heard two main groups of notes: the first four notes, grouped together, followed by the next four notes, also grouped. Not only has the listener formed a mental boundary between the two groups, but also a connection between them has been established. In short, an analogy is perceived between the groups (Figure 1.3).

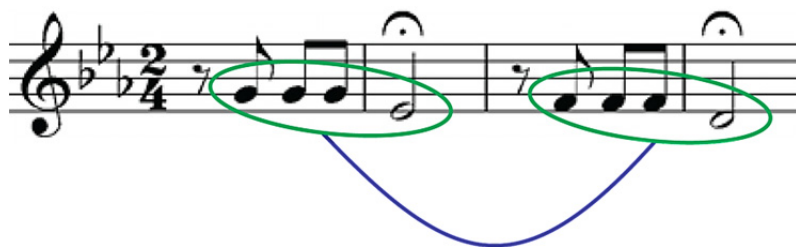


Figure 1.3: Previous figure with groups and an analogy.

We see that there are two groups of notes, surrounded by ellipses, and there is an analogy between the groups, indicated by an arc. This is the type of mental representation modeled by Musicat. The program’s goal is to *simulate real-time listening by creating internal representations (such as groups and analogies) of the musical structures it “hears” as music is given to the program one note at a time.*

Wherefore Musicat?

But why, one may ask, is modeling music-listening an interesting problem? The example above looks very simple. And isn’t listening a passive activity? People listen to music all the time, without apparent effort. These days, computers are supposed to be great at recognizing patterns, as well — this is the era of statistical machine learning and “Big Data”, after all! — so can’t we just run an off-the-shelf “pattern recognizer” on music and be done? Smartphones can already “listen” to and identify songs on the radio. So, what is it about how *people* listen to music that makes it worthy of study?

I argue that listening is a much more active process than many people give it credit for. There is much more to listening than meets the ear, as I will now explain.

MUSIC-LISTENING AS PERFORMANCE

Listening might seem so passive that one might expect it to be simple (or pointless) to model. What would be the point of a computer that didn't do anything but sit on a desk and listen to songs? I will make three claims about listening to show that it is a more active process than most of us probably realize:

1. Music-listening is dynamic.
2. Music-listening is a creative process.
3. Music-listening is not a universal talent.

In order to emphasize that music-listening involves cognitive work, I introduce the term “listening performance” to refer to the *creative* act of listening.

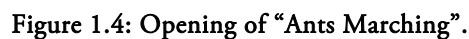
MUSIC-LISTENING IS DYNAMIC

“To stop the flow of music would be like the stopping of time itself, incredible and inconceivable.” — Aaron Copland (Copland, 1960)

Music-listening is a temporal process. Whereas we can see an entire painting at one time, music comes to us as a *series* of notes, with new notes coming into our conscious awareness as older ones quickly fade away. Just this temporal aspect alone might make one think that music requires focused attention in listening. Music is even more dynamic than this suggests, however.

One might at first be tempted to think of serious, focused listening as follows: it is a mental process that registers each new note or chord as it is sounded, dutifully paying attention to the passing music in the same way that a copy machine or scanner passes a light over a piece of paper and transfers the original to the copy, one horizontal slice at a time. After all, this is how musicians perform music — one note after another, until a piece is

Example: Ants Marching



Snare Drum

Fiddle

Figure 1.5: Opening melodic motif in “Ants Marching”.

The drumbeat happens again, and in the next measure, there are three more chords in the fiddle, with a simple melody on notes F# and G in the lower voice. At about this point for me, mental confusion ensues and I feel disoriented as I reinterpret the metrical structure. A moment later, the rhythmic structure suddenly falls back into place for me and I re-hear the music in the way notated in Figure 1.6, with the drum pattern shifted to the *offbeats*:



Figure 1.6: Rhythmic reinterpretation of Figure 1.5: drum on beats 2 and 4, not 1 and 3.

Intellectually, I now realize I had been “tricked” the whole time by the drum, which started the song a minute earlier by playing on an offbeat, not the downbeat, as I had assumed. It feels, however, that something “shifted” in a hazy, disorienting way as the first notes of the actual melody were playing. Specifically, I had to reinterpret and rehear several drum attacks, even including a few that occurred before the fiddle’s entrance, as falling on beats 2 and 4, not 1 and 3.

This last phenomenon is what I mean when I claim that music-listening is a dynamic and retroactive process. Mental interpretations of musical structures are not static, and may even change retroactively, after they have been formed. Although we can’t go back in time, we do hold recently-perceived music in working memory for a brief time, or the order of seconds (perhaps by using a mechanism known as the *phonological loop* that allows several seconds of music to be heard in the “perceptual present”— see Chapter 2), and we can

indeed go back and revise our perceptions of chunks of music that we have just heard and that is still in working memory.

MUSIC-LISTENING IS A CREATIVE PROCESS

“To listen is an effort, and just to hear is no merit. A duck hears also.” — Igor Stravinsky

Implicit in the discussion in the previous section is the idea that music-listening involves *creating*. Specifically, perceptual structures representing what we have heard are formed in working memory. These are mental objects, not physical and tangible structures, but they are no less real. In this section I elaborate on this idea, starting with some commentary on a relevant-sounding but quite differently motivated project.

Some Thoughts on David Cope’s EMI

Any mention of creativity, music, and computers is likely to bring to mind the work of David Cope, so a few words at the outset will be useful to provide some context and to contrast his approach with the one motivating the design of Musicat. Cope’s computer program EMI (*Experiments in Musical Intelligence*) was an ambitious and thought-provoking attempt to make a computer compose music in the style of human composers (Cope, 1996). EMI generated pieces of music that sound in many ways like Bach’s inventions, Chopin’s mazurkas, or Mahler’s symphonies — enough so that many musically-educated listeners have had trouble distinguishing human-composed pieces from EMI’s imitations. What, then, is left for the science of creativity, if computers can already churn out symphonies in droves? The fact is, though, that incredible challenges remain — and one of Mozart’s hobbies, the *Musikalisches Würfelspiel* (musical dice game), provides a clue to what is missing from EMI.

Popular in the 1700's, the musical dice game was a simple algorithmic way of generating musical pieces: dice were used to select pre-composed segments of music at random, which were then strung together in sequence. Many composers of the time (Mozart, for instance) enjoyed composing short segments of music to be used in the process, carefully ensuring that the end of each segment would flow nicely into the beginning of any potential successive segment. Once the short segments had been written, the dice were rolled, fragments were randomly selected, one by one, and the final piece was written down. It is obvious that any creativity one might attach to the result derives from the composition process that occurred *before* the dice-rolling started.

EMI may be seen as using existing human-composed music as input to its own musical dice game — albeit one with quite elaborate rules for combining segments. Roughly speaking, EMI functions as a sophisticated machine for chopping up preexisting music into small fragments and then coherently stitching together those fragments into a larger structure (Cope & Hofstadter, 2001). Indeed, Cope himself repeatedly likens his program to the *Musikalisches Würfelspiel*. Listeners who are very familiar with the source compositions by human composers can often hear original fragments stitched together in EMI's output. (In certain of EMI's pieces, it is far subtler to hear the influence of the source materials, and even highly trained musicians have been fooled into thinking that EMI's output was human-composed.) The computer, however, has no awareness of its own musical output and doesn't even notice if it plagiarizes an entire phrase of a Bach invention — it has no concept of creativity, no mechanism to distinguish exciting new ideas from mere quotations, no inner drive towards genuine novelty, no inner emotional fire. Although it generates music, EMI does so in a manner that seems to me to be radically different from how humans compose. Admittedly, Cope himself might disagree: see (Cope, 1996) for his point of view.

The research program motivating the design of Musicat is committed to the notion of coming to an understanding of human creativity not by frontally attacking the gigantic high-level problem of how a gifted composer dreams up a nocturne or a symphony, but instead via a more humble type of modeling of the fundamental mechanisms of genuine human perception and thought.

Some Thoughts on the Scope of This Project

This research aims to probe mechanisms of creative thought that are fundamental both to music cognition and to cognition in general. Instead of attempting to model the enormously complex process of listening to large works of music or to deal with issues of cross-cultural music perception, this project restricts the modeling task to a particular subclass of music (melodies with simple harmony in the Western tradition). The model proposed here approximates the cognitive processes that take place when a person listens to short, simple-sounding fragments of music. This restriction can be surprisingly helpful — not only does it bring the modeling task into the realm of the tractable, but it also forces the model to focus on core perceptual issues.

Some Thoughts on Creative Music Perception

Perception is an active process in which the brain interprets sensory stimuli, such as sound waves, and transforms the raw input into new mental structures that are more suitable — indeed, essential — for the activities of high-level cognition. Far from being a passive, mechanical algorithm hard-wired into our brains from birth, this process of interpretation is dynamic, involving a combination of bottom-up and top-down processing. By necessity, the interpretive process ignores a huge number of details and focuses attention on the most

important aspects of a situation — otherwise, we would be overwhelmed with sensory information (just think of the huge amount of detail available to our senses in, say, a crowded restaurant: the number of objects we can see, things we can smell, or sounds we can hear is nearly uncountable). Such perception is subjective and creative. A central tenet of the philosophy underlying this work is that exactly the same processes that give rise to the high-level *generation* of creative thoughts, ideas, and works of art also drive the creative *perception* of sensory input. The notion of creativity residing in the humble act of perception might seem, at first glance, far-fetched, but the view advocated here is that the essence of creativity is evident both in “mere” perception and in activities that we standardly think of as creative.

Creativity is an intimate part of music perception in particular because listening to music requires a listener to form subjective internal mental representations of the music. Two different people listening to the exact same recording of a piece can have radically different experiences, as if they each heard completely different music — consider a European hearing Balinese music for the first time, and a Balinese person hearing the same piece. The raw sensory input is the same for both listeners, but their subjective experiences and their internal representations of the heard music would be quite different. We need not go as far as Bali to see how listeners differ, however. Even two American listeners will likely hear the same piece in quite different ways: a person who loves romantic-era music will certainly understand Chopin in a different way from a person who loves heavy metal. Furthermore, a classical musician familiar with baroque music might hear Chopin differently from the way a musician familiar with romantic music would. In the extreme case, even two different professional concert pianists may hear the same Chopin piece differently — just think of how different two different performers’ interpretations of the same piece can be. Although performing and listening are distinct talents, I suspect that some of the variability in

interpretation stems from the variability performers' own idiosyncratic ways of listening. Generating representations is a creative act, as is demonstrated by the uniqueness of each listener's experience. These internal representations also induce expectations about what the music will do next. This act of generating expectations is closely related to the process of music composition, and is obviously creative.

MUSIC-LISTENING IS NOT A UNIVERSAL TALENT

“When I speak of the gifted listener, I am thinking of the non-musician primarily, of the listener who intends to retain his amateur status. It is the thought of just such a listener that excites the composer in me.” — Aaron Copland

I have argued above that music-listening is a dynamic and creative process. In this section I go further, claiming that people have differing degrees of talent for listening. Musicat's listening is extremely primitive compared to almost all human listening, but one should bear in mind that Musicat's ultimate goal is to model the listening performance of what Copland called a “gifted listener” in the quote above. Otherwise, we could imagine that much simpler programs could possibly model “listening”.

We are surrounded by music in contemporary society. I am listening to music on my headphones as I write these words. We routinely listen to music streamed in real time from the Internet. We can carry countless of gigabytes of music on our iPods to listen to as we go jogging or driving. We can't avoid the nearly constant stream of background music playing in stores or restaurants. Not long ago, I was in a restaurant where the music from a live band in one room was competing with prerecorded music that was playing in the next room. What a cacophony! And yet such things are quite commonplace today.

Inundated with music as we are, we might be inclined to think of everyone as an expert listener — with such intense exposure to music all the time, it would seem plausible that we'd all be really good at listening. However, the kind of listening that I am interested in is not automatic, nor is it even possible for everyone. As an analogy, consider photography. In the early days, people thought that photography involved simply clicking a button to take a picture — the camera would do all the work. In contrast, in the case of arts such as drawing or painting, the artist was clearly putting in a lot of effort. But as we understand well today, photography too is an art, requiring thoughtfulness, talent, and work, although the work of a photographer might be less evident as such than that of a painter. A photographer is also *creating* by composing the scene, selecting the particular position of the camera, arranging for the right lighting, making choices of lens and aperture, setting the focus on a particular part of the scene, and so on and so forth. The photographer, above all, is thoughtfully considering the visual scene and making choices about how to transmit a particular view of the scene to another person through an image. Just as the photographer does more than simply click a button to take a photo, the listener must actively consider music, decide which aspects of the music to focus on, and create internal representations of the music that is heard — it is much more than simply clicking a button or putting on a pair of headphones.

Listeners perform little externally-apparent “work”, but people differ greatly in their skill at listening. Some years ago, it was not uncommon to hear a person modestly say, “I have a tin ear” or “I’m tone-deaf” to indicate a lack of music-listening skill. Although these phrases have somewhat fallen out of fashion, it stands to reason that people today still vary in their levels of listening skill.

Listening skill varies among people, because listening is quite complex. I want to be clear up-front that Musicat is not a great listener when compared with most people. The program, as will be seen, still fails to understand some things that almost all human listeners

(at least those who grew up listening to Western tonal music) would intuitively understand. At the same time, though, it “hears”, with its “silicon ears”, a few rather sophisticated things that might go unnoticed by the tone-deaf “tin ears” among us.

Overview of this Dissertation

This dissertation could be seen as consisting of four parts, as follows:

1. Four initial chapters (including this one) set Musicat in the context of other work, explain the domain of music-listening that Musicat is concerned with, and discuss why analogy-making is central to this work.
2. The next three chapters give numerous detailed examples of Musicat in action, showing its listening performances on a variety of melodies.
3. The next two chapters describe the design of the Musicat program, both in its present form and in earlier incarnations.
4. The final chapter includes a discussion of Musicat’s current state as a listener, describing what it does well and what it does badly, and suggesting directions for future work.

There are also three appendices providing bibliographic references, a description of a pilot study, and some preliminary quantitative results comparing Musicat to another model.

Whither Musicat?

In his Norton Lectures in 1973, Leonard Bernstein asked the question “Whither music?” and more specifically “Whither music in our time?” For Bernstein, this was a crucial question — it was his version of Charles Ives’ musical “Unanswered Question”.

I will give some tentative and speculative answers to my own question “Whither Musicat?” at the end of this thesis, in Chapter 10. But for now, harkening back to Ives, I’ll leave the question unanswered. It is hard to say where Musicat is headed, because Musicat still has a long way to go before it will come anywhere close to the complexities of human listening; today’s version is just a tiny first step towards understanding listening as done by people. But it’s a step in the right direction, in my extremely biased opinion.

CHAPTER TWO

Work that has Influenced this Project

Models of Melodic Expectation and Cognition

LEONARD MEYER

Emotion and Meaning in Music (Meyer, 1956) is the foundation of most modern work in music cognition. Based on gestalt psychology and the idea of musical expectation but also delving into the emotional nature of musical listening, Meyer's work is far-ranging and full of interesting ideas. I present just a few highlights relevant to my work.

Emotion and Meaning

Meyer states that the “central thesis of the psychological theory of emotions” is that “Emotion or affect is aroused when a tendency to respond is arrested or inhibited.” (p. 14) That is, when a given stimulus occurs, a person may naturally respond by reacting in a certain way or having certain thoughts in response. In music, a simple example is the stimulus of an ascending scale that stops on the leading tone, just short of the tonic. The ascending scale evokes the mental response of expecting the tonic to occur next. However, when the expected tonic does not occur, the listener will have an emotional response (in this case, frustration). According to Meyer, emotion derives from this sort of response-inhibition.

He discusses how once a listener is accustomed to the conventions of a particular musical culture, tension arises when expectations are not fulfilled; “deviations can be regarded as emotional or affective stimuli.” (p. 32)

On meaning, Meyer carefully notes that in many forms of human communication, meaning is about how one stimulus (such as a word or text) can refer to another item (such as a concrete object or an event or action). He terms this *designative* meaning. However, he points out a special situation where the stimulus item can refer to another thing of the very same type as the stimulus itself; his example is that the dim light of dawn foreshadows the arrival of the full sunlight of the day. He calls this case *embodied* meaning, and notes that this type of meaning is especially important in music (especially “absolutist” music). For Meyer, embodied meaning in music is where one musical event causes the expectation of another musical event. “Embodied musical meaning is, in short, a product of expectation” (p. 35). If a stimulus causes an expectation, “then that stimulus has meaning.”

Interestingly, Meyer attributes both emotional response and musical meaning to expectation. Expectation comes from two sources according to Meyer: from learned patterns in a particular musical style as well as from innate perceptual processes as described by gestalt psychology. While Meyer is a proponent of these gestalt ideas, he also is careful to say that “any generalized gestalt account of musical perception is out of the question” because of the influence of style.

Gestalt Principles

I list a few of Meyer’s applications of gestalt principles to music here, as they are critical in my work. The Law of Prägnanz is an overarching idea about how people naturally form mental representations that are as well-structured and concise as possible. The Law of

Good Continuation describes how musical processes are expected to continue, so it plays a key role in modeling expectation. The Law of Closure, on the other hand, describes how we hear some structures as completed, which is important in modeling grouping.

The Law of Prägnanz

According to the Law of Prägnanz, people strive for concise, simple, symmetric, well-structured mental representations of the things they perceive. I think of this as a sort of Occam's Razor of perception. Occam's Razor states that all other things being equal, we should prefer the simplest explanation that fits an observed situation. The Law of Prägnanz, similarly, says that people naturally desire simple mental representations. This mental striving for simplicity leads to expectations for structures (visual shapes, observed physical motions, musical gestures, and so on) to continue on to form a complete structure that is easy to represent as a whole. The Law of Good Continuation is a natural consequence of this idea.

The Law of Good Continuation

The Law of Good Continuation (p. 92) states that we expect patterns to continue in the same way, once they are started. Meyer defines several terms which were used very frequently later on (especially by Eugene Narmour in his Implication–Realization model), including *process continuation* and *process reversal*. Process continuation (or simply continuation) refers to the normal mode of “continuing in the same way”. For instance, a melody ascending through a major scale is an example of process continuation — at any point until reaching the tonic, we may expect the scale to simply continue. Harmonic motion around the circle of fifths is another example. On the other hand, process reversal, or simply reversal, refers to the stopping of a process. For example, a melody that ascends through the scale from one tonic to the next, an octave above, undergoes a process reversal if

the high tonic is held, thus stopping the ascending motion. Note that a reversal in direction is not required; it is the ending of the continued process that reversal refers to, even if the melody does not change direction and move downwards. Process reversal is an abstract type of reversal, unrelated to surface-level reversing of direction.

Meyer discusses several types of continuation, including melodic and rhythmic continuation. Some examples of melodic continuation are:

- An ascending or descending major scale.
- Notes ascending or descending through a triad.
- A phrase that ends on a particular note, followed by a phrase that starts on that same note. The third phrase would be expected to start with the final note of the second phrase if this is heard as a process.

Some examples of rhythmic continuation are:

- Perception of equal pulses (*e.g.*, as described by “internal clock” models — the section on Povel and Essens, later).
- Perception of meter (accented hierarchical beat structure).
- Perception of rhythm: “grouping one or more unaccented beats in relation to an accented beat” (p. 103).

Rhythmic perception depends critically on which notes are heard as accented. Meyer points out how *dynamic stresses* (*i.e.*, stresses added to notes by a performed modifying the notes’ volumes or articulations) are not necessarily the same as *perceptual accents*, which are mental constructs based on a listener’s perception: “Basically anything is accented when it is marked for consciousness in some way. Such mental marking may be the result of differences in

intensity, duration, melodic structure, harmonic progression, instrumentation, or any other mode of articulation which can differentiate one stimulus or group of stimuli from others. Even a silence, a rest, may be accented..." (p. 103).

Completion and Closure

The Law of Prägnanz suggests that the mind is "continually striving for completeness and stability and rest." (p. 128) Meyer gives several concrete ideas about how a musical structure might sound completed. These ideas about grouping have influenced many models of music grouping, including that of Musicat, so it is helpful to review them here. I focus on Meyer's discussion of tonal organization and melodic shape; he also mentions rhythmic completeness and harmonic completeness, but has less to say about them. (In brief, harmonic completeness arises from motion to the tonic, as we would expect, whereas rhythmic completeness has to do with the "apprehension of a relationship between accented and unaccented parts of a cohesive group." (p. 143)

Meyer writes: "In any particular musical work certain melodic patterns because of their palpable and cohesive shapes become established in the mind of the listener as given, axiomatic sound terms." What is it about certain shapes that make them sound complete? For Meyer a primary force is the tonal structure in the musical culture of the piece in question. In Western tonal music, the tonic note typically provides closure, especially when approached stepwise. In other cultures, a different melodic structure might provide similar closure. Even the repetition within a piece of a melodic segment that normally would not sound complete can cause a listener to hear the segment as establishing closure. In addition to repetition and tonal structure, Meyer discusses melodic direction, but his focus is on expectations instead of on completeness. (Perhaps this is because he was implying that if a melody moves as expected it sounds complete.) Although he uses different terminology,

Meyer essentially says that in pitch height we expect melodic regression to the mean — in other words, once a melody moves to a high or low register (taking into account the melodic range of a particular instrument or voice) we expect a melody to eventually return to the middle of its vertical range. Listeners hear melodies proceeding as expected as moving towards closure, while melodies denying expectation will sound incomplete.

More generally, Meyer goes beyond gestalt principles and suggests that closure is also due to the shape of tension and relaxation in a melody. For example, when a melody descends, it sounds more relaxed and leads to a sense of closure. More generally still, he asserts that “completeness is directly related to our ability to understand the meaning of a particular pattern” (pp. 138–9). A pattern that is not understood in any special way and that causes no sense of “process” will remain incomplete. Completeness is the result, then, of setting up expectations and motion, and then fulfilling the expectations somehow and coming back to a stable state. This general principle applies to Meyer’s ideas not only of melody, but also of rhythm and harmony.

EUGENE NARMOUR

Overview

Narmour’s Implication-Realization (I–R) model focuses on the implications (*i.e.*, expectations) generated by the movement from a note to another note and the ways in which the implications are realized (*i.e.*, come to pass as expected) or denied. The model is ambitious in scope, encompassing such topics as bottom-up and top-down perception, the effects of intra- and extra-opus style, and the multi-parametric nature of melody. A novel

taxonomy is introduced (Narmour's "genetic code") for categorizing the types of implications and realizations in a melody.

Gestalt Principles and the Problem of Style

The I–R model is primarily concerned with implications generated by subconscious, bottom-up processing. However, Narmour goes to great lengths to explain the important top-down influence of style. In broad terms, he states that for a listener the features of the input music result in subconscious expectations *except* when the context of style modifies those expectations. The roles of top-down and bottom-up processing are clearly separate for Narmour, who insists on "the existence of two expectation systems. The top-down one is flexible and variable but controlled; the bottom-up one is rigid, reflexive, and automatic – a computational, syntactic input system." (Narmour 1990, 54) Although Narmour is careful to put top-down and bottom-up processing on an even footing, the majority of the theory, including a system of symbols introduced used to describe any melody in terms of implications and realizations for various musical parameters, focus on bottom-up expectations rooted in gestalt principles. Style is not treated analytically in the way that note-to-note details of melody are, but Narmour does expose many inherent difficulties with incorporating style into a theory of expectation.

Interestingly, the I–R model states that both top-down and bottom-up processing occur at multiple levels of musical hierarchy. The "top" and "bottom" in "top-down" and "bottom-up", then, must not be confused with "surface" and "deep" structures of musical hierarchy. This theme — the application of the same principles at several hierarchical levels — shows up in several models, including Lerdahl and Jackendoff's *Generative Theory of Tonal Music* and Larson's Seek Well model, as well as in Schenkerian analysis. However, the cognitive basis for applying gestalt principles to musical events at deeper levels (*i.e.*, with a

longer time span between events) may still need to be tested. Specifically, if low-level processing describes an innate universal mechanism for processing *raw* perceptual input, it remains to be shown that it is cognitively plausible for this same mechanism to be applied to a more *abstract* level of input (*e.g.*, a background structure in a Schenkerian analysis) that has been extracted from the raw input.

In the study of bottom-up expectations, Narmour invokes the gestalt principles of *similarity*, *proximity*, and *common direction* as cognitive universals that apply to individual musical parameters such as the direction of pitch motion and the size of pitch motion (in terms of vertical intervals between pitches). Additionally, he proposes two new hypotheses — *reversal* and *parametric scale* — to supplement those three gestalt principles. All in all, there are five specific aspects of melody used in classification and description in Narmour's system. These aspects are:

1. Registral direction
2. Intervallic difference
3. Registral return
4. Proximity
5. Closure

Five Fundamental Principles

Registral direction

Registral direction refers to the change in pitch from one note to the next in a melody. The direction is either up, down, or lateral (no change). As long as the interval between two pitches is small, this principle states that a listener will subconsciously expect a

continuation of pitch direction: upwards, downwards, or lateral motion is expected to continue. A large interval, on the other hand, will lead to an expectation of *reversal* of direction.

An interval is defined to be *small* if it is less than a tritone and *large* if greater than a tritone — the tritone interval itself is a boundary case. Narmour sometimes uses this classification of intervals into these distinct categories in his writing, but he makes it clear early in his book that perception of interval size really operates on a continuum; these categories are used sometimes for convenience but must not be taken too seriously (it seems to me that people often forget about these disclaimers that Narmour makes). Thus, the I–R model’s prediction of either a continuation or reversal is not as simple as the interval-size metric based on strict categories might suggest. Narmour uses the term “parametric scale” to refer to a continuum (or an ordered set) of possible values that is psychologically innate — perception of the parameter involved on a particular parametric scale is subconscious and low-level — and automatically generates implications. Narmour writes that interval size and the strength of resulting implications can be considered as operating along a parametric scale. A very large interval on this scale generates a stronger implication of reversal than a moderately large interval; the same holds for very small intervals and implication of continuation. Additionally, even a very large interval may result in a so-called “recessive implication” for continuation. The word “recessive” means that in retrospect, a listener may reinterpret previous parts of a melody as having implied the notes that actually occurred, even though the implication was quite different earlier in the listening process. For example, a leap up of a major fifth might result in a strong implication for process reversal (*e.g.*, for the melody to move down after the large leap). Narmour calls this the “dominant implication”. However, if the leap were to be followed by another leap instead — say, up again by a perfect

fourth — the listener might retroactively hear this second leap as a realization of a recessive implication for process continuation.

Intervallic difference

While registral direction refers to the direction of motion between pitches, intervallic difference relates to the size of the interval between pitches. (The size of interval was involved in registral direction in determining the strength of the implication, as discussed in the previous section, but the size of an interval itself is also a parameter that can be implied.) The principle states that small intervals imply a continuation with similar-sized intervals. Large intervals, however, imply relatively a continuation with smaller intervals. Large and small intervals were already defined above in reference to registral direction, but here the concept of “similar-sized” intervals is also important. Narmour defines interval similarity as a difference of a minor third or less in interval size.

Registral return

The typical motion of a melody away from a pitch and then back to the original pitch is known as registral return. Exact registral return describes the perfectly symmetrical melodic archetype **aba**, exemplified in typical melodic shapes such motion to and from a neighbor tone (*e.g.*, the melody G–A–G over a G-major chord). Near-registral return (**aba'**) occurs when the final tone is very similar to the original pitch (within two semitones). Patterns become less archetypal as they deviate more from the symmetrical case.

Recognition of registral return can make an otherwise surprising realization more expected. This idea is also used to explain melodic “streaming” in the I–R model. Streaming is the process by which a monophonic melody is heard as consisting of two or more

simultaneous melodies, generally separated by register (pitch height). This occurs often, for example, in Bach's Suite for unaccompanied cello — a melody and countermelody are interleaved, with the cello quickly alternating between low and high notes. In this case, every three notes involves near-registral return. According to Narmour, the listener hears such a melody as a series of overlapping **aba'** processes, which aids the listener in splitting the music into multiple input streams in different registers (Narmour, 1992, p. 352).

Proximity

When the gestalt notion of proximity is applied to pitches, the result according to I–R theory is that small intervals are more strongly implied than large intervals. Additionally, the implications generated by larger-sized intervals are said to be stronger than for relatively smaller intervals.

Closure

Closure describes how listeners break up melodies into separate perceived segments. According to the I–R model, closure in the pitch domain occurs in two cases:

1. The melody changes direction.
2. A relatively larger interval is followed by a smaller interval.

Naturally, parameters other than pitch also can give rise to a feeling of closure. Narmour lists the others as:

1. Interruption of an implied pattern.
2. Strong metric emphasis.
3. Dissonance resolving to consonance.
4. Short notes moving to long notes.

I–R Symbols

The I–R model annotates melodies with symbol strings based on the implications and realizations present in terms of the parameters of registral direction and intervallic difference. Narmour’s 16 basic symbols (see his book for details) typically apply to groups of three notes: the first two notes set up an implication based on the pitches and the interval between them, while the third note produces an interval with the second note; this interval and the third pitch may realize or deny aspects of the implication of the first two notes. A sample string describing a melody is “ID-IR-VP-P-IP-VR-D-R-M”. The I–R theory hypothesizes that its symbol system can be used “to represent the listener’s encoding of many of the basic structures of melody” (Narmour, 1990, pp. 6–7).

Tonal Pitch Space and the I–R Model

Narmour’s focus is on innate, bottom-up gestalt laws, so it is natural to wonder how the concept of tonal scale step fits in to the theory. For example, although C-G-C# and C-G-D are both examples of near-registral return, the former seems less expected if a C-major context has been established. The I–R theory explicitly states that it supplements conventional notions of tonal pitch space and provides another dimension to pitch relation that should be considered. This is a case where a listener’s understanding of musical style interacts with bottom-up perception. Rather than defying accepted notions of pitch space, Narmour intends to contribute to a more complete account of pitch perception.

After making a point of separating the function of scale steps in harmonic vs. melodic contexts, Narmour introduces several categories of scale steps in order to introduce a parametric scale (recall the description of this term above) for melodic implication with respect to scale step. Degrees 1, 3, and 5 are called *goal notes* (GN), degrees 2, 4, and 6 are

nongoals (NG), and the leading tone is a *mobile note* (MN). The I–R theory already states that small intervals imply continuation while large ones imply reversal. Taking the type of scale step into account affects the strength of the generated implication. Within a clear tonal context, the more differentiated the two tones of an interval are with respect to the scale step categories above, the stronger the implication. Thus an interval moving from 7 to 1 (MN to GN) generates a stronger implication than 2 to 4 (NG to NG) or 1 to 3 (GN to GN). Narmour enumerates all nine possibilities: the most open (i.e. most implicative) combination is GN to MN, while MN to GN is the most closed. Within each category, the particular scale degree chosen also affects the generated implication strengths.

Narmour does not place much emphasis on tonal pitch space in his model, and states that because tonal style is learned, it deserves no more preferential treatment than other parameters affected by learning (Narmour, 1992, p. 85). The focus of the I–R model is on bottom-up processes.

Summary

The I–R theory presents a wealth of ideas about low-level music cognition, especially on the note-to-note level. Although some researchers have shown that parts of the theory, particularly those involving symbol strings and their implications, may be simplified without loss of predictive power (Schellenberg, 1997), I think that these sorts of tests have oversimplified the theory, especially since Narmour goes to some length to discuss the complexity and context-dependent nature of many aspects of melody. Other criticism seems well-founded, however. For example, Margulis (2005) provides a useful critique of the theory that points out how the I–R symbols for basic melodic structures provide taxonomic categorization but do not clearly explain the expectations generated by a melody. Similarly, the question of how expectation relates to affect is mostly ignored by the theory. Finally,

melodic hierarchy is invoked by Narmour but the focus remains on local note-to-note relations. Despite shortcomings of the theory, however, I find many of Narmour's comments on the interaction between top-down and bottom-up processing to be especially useful.

ADAM OCKELFORD

Repetition of musical structure is also important to Adam Ockelford (Ockelford, 1991). To describe low-level perception, he coins the term *perspects*, a contraction of the phrase “perceived aspects”, to describe parameters of a note that have been perceived by a listener, including such qualities as pitch, duration, volume, timbre, attack time, etc. Larger groups of notes are also associated with such *perspects* as key and meter. Ockelford describes how a listener may perceive the relationship between certain aspects of the *perspects* attached to two different notes (for example, a note heard as a quarter-note might be followed by a note heard as a half-note; this relationship would be heard as a doubling in the *duration* *perspect*.) Next, he presents the concept of higher-order relationships perceived between relationships themselves (meta-relationships). Finally, he discusses *zygonic relationships* between *perspects*, where the temporal order of events is critical and where a given *perspect* can influence how. The curious word “zygonic” derives from the Greek word for “yoke” and is used by Ockelford to indicate a relationship in which two things are linked in a particular fashion: typically the first (earlier in time) musical object in the relationship is heard as having an effect on how a later object is perceived. (Sometimes these relationships are heard in the other direction — that is, retrospectively — just as was the case with Narmour's “recessive” implications.) Ockelford's stresses the importance in the directionality of the flow of time when discussing these *zygonic relationships*. Additionally, although Ockelford does

not talk specifically about analogy, it seems that zygonic relationships have much in common with the notion of musical analogies described later, in Chapter 4.

OLIVIER LARTILLOT

Each of the authors mentioned in the three previous sections — Meyer, Narmour, and Ockelford — wrote in some way about the importance of recognizing repetition in musical structure. Olivier Lartillot’s has developed a computer library named “kanthume” (later called “OMkanthus”) (Lartillot, 2002, 2004) that looks for musical patterns in a cognitively-motivated manner that, like Musicat, is explicitly concerned with analogy-making and real-time perception:

“...an analogy is the inference of an identity relation between two entities knowing some *partial* identity relation between them. Our discussion about the phenomenon of time in music leads to the idea that our experience of music is a temporal progression of partial points of view that consists of the *timely local* perception of it. Thus the analogy hypothesis of music understanding means that the global music structure is inferred through induction of hypotheses from local viewpoint. [sic]” (Lartillot, 2002)

(The term “viewpoint” refers to musical parameters that are the focus of attention in a particular context — this seems quite similar to Ockelford’s “perspects”.) Even though Musicat and kanthume have completely different architectures, the projects have similar guiding philosophies.

The kanthume program takes symbolic musical scores as input. It groups notes in the score together and detects repeated motivic fragments in a piece, forming a motivic network that is implicitly stored in long-term memory (the program shows the network as a collection of rectangles and relationships superimposed on a musical score). The program discovers relationships between groups: it can notice that a group is a transposition of another group, or that a rhythm of a group is equivalent to that of another group where all duration have

been cut in half (*i.e.*, it recognizes augmentation and diminution of rhythms). The model finds such relationships by attempting to determine which parameters are the most important in a given context, and then it extracts motifs based on intervallic, contour, and rhythmic relationships as appropriate. The program uses heuristics to prune away irrelevant candidate motifs.

Lartillot's model makes a special effort to model the effect of temporal relationships. On human music cognition, Lartillot writes, "a pattern will be more easily detected if it is presented very clearly first, and then hidden in a complex background than the reverse." The computer model is designed to respect temporal ordering during listening, and, much like Musicat does, holds notes in a simulated short-term memory buffer. Certain links between notes explicitly account for temporal ordering: Lartillot uses the term *syntagmatic relations* to refer to links between notes that form a temporally-directed relationship in which one note of a motif stored in memory can activate the memory of the successive note. (To me, Lartillot's "syntagmatic relations", when taken in conjunction with his notion of "viewpoints", seem quite similar to Ockelford's "zygonic relationships".) In other words, these relations represent memory items that are linked in a temporal sequence such that the first note of a sequence can cause the second, and then the third, and so forth to be recalled from memory.

The reader may be curious at this point to know how Musicat and kanthume differ, because both projects involve real-time listening and analogy-making. One major difference is that Musicat's highly-stochastic architecture is based on a simulation of various internal forces and pressures (some of them working in concert and others in conflict) that generate various grouping structures and relationships, in what is often a frenzied and chaotic-looking manner, but from which a stable structure gradually emerges. Kanthume, on the other hand,

takes a more traditional algorithmic approach, using heuristics with names such as “maximization of specificity” and “factorization of periodicity” to find a satisfactory parsing of a musical fragment in a deterministic way.

FRED LERDAHL AND RAY JACKENDOFF

A Generative Theory of Tonal Music, or GTTM (Lerdahl & Jackendoff, 1983) has arguably been the most influential single work on music cognition in the past three decades. The book was inspired by Leonard Bernstein’s lecture series at Harvard (Bernstein, 1973), in which he made an analogy between Noam Chomsky’s ideas about a universal grammar in linguistics and a possible universal grammar of music. GTTM was a formal attempt at describing a new system of music analysis based on linguistic ideas such as formal grammars. Rather than attempting to summarize this large and complex work, I will point out several ideas in GTTM that are relevant to my project. Particularly interesting are the chapters on grouping and meter and the general notion of combining two different types of rules (*well-formedness rules* and *preference rules*). GTTM analyses are visually striking due to the two types of grammar-like tree structures created in accordance with the rules of the theory, so I will also touch on *time-span reduction* and *prolongational reduction* trees.

Well-Formedness and Preference Rules

GTTM provides notation for analyzing several aspects of music such as hierarchical grouping and tension–release structure. For any piece, a formal “analysis” of its structure can be generated using the rules and guidelines in GTTM. *Well-formedness* rules prescribe a specific “mathematical” form for any acceptable analysis; any analysis according to the theory must follow these rules precisely. *Preference rules*, on the other hand, suggest how to choose between competing analysis possibilities. While well-formedness rules are analogous to rules

in a formal linguistic theory, preference rules are a unique contribution Lerdahl and Jackendoff found necessary for describing *musical* grammar. They note that:

the interesting musical issues usually concern what is the most coherent or “preferred” way to hear a passage. Musical grammar must be able to express these preferences among interpretations, a function that is largely absent from generative linguistic theory. Generally, we expect the musical grammar to yield clear-cut results where there are clear-cut intuitive judgments and weaker or ambiguous results where intuitions are less clear. (p. 9)

Temperley (see below) produced a concrete computer model for several aspects of musical cognition based on some of the preference rules in GTTM. The operation of some of the “codelets” in my project can also be seen as an alternate mechanism for implementing some of these preference rules.

Grouping and Meter

GTTM begins with a description of hierarchical grouping structure of music. Interestingly, pitch is absent from the initial chapter. Core concepts include the following:

- Musical groups combine in a hierarchical fashion.
- The hierarchy is nearly strict; groups do not overlap except for very short segments where the end of one group can be elided with the start of the successive group.
- The pattern of accents at the musical surface gives rise to an implied metric hierarchy consistent with the pattern.
- Grouping structure and metrical hierarchy interact with each other, but are distinct and should not be confused; they be or may not be in phase with each other.

Grouping structure is constrained by well-formedness rules that formalize how groups form hierarchies. Preference rules that choose between alternate well-formed group structures are derived largely from gestalt psychology ideas. Some of the rules (paraphrased) include:

- Avoid analyses with very small groups.
- Locally-large rhythmic gaps maybe heard as group boundaries.
- Larger rhythmic gaps indicate group boundaries at higher hierarchical levels.
- Prefer analyses where groups are subdivided into two equally-sized pieces.
- Parallel segments of music should form parallel groups (see Chapter 4 for more discussion about the ambiguity of the word “parallel”).

Metrical structure is likewise described by both types of rules. The well-formedness rules here state rather obvious things such that each beat at a level must also be a beat at lower hierarchical levels, but also less obvious things such as:

- Strong beats at each level are two or three beats apart.
- Each level must consist of equally-spaced beats.

The preference rules for meter include (paraphrased):

- Prefer structures where strong beats appear early in a group.
- Prefer strong beats to line up with note attacks.
- Prefer strong beats to correspond with stressed notes (stressed notes are those performed with an articulation or change in dynamics that distinguishes them from nearby notes).

These rules are reminiscent of Povel and Essen's rules (1985) for internal clock induction (described in the section on Musical Rhythm below).

Reduction Trees

GTTM analyzes music using tree structures called *reductions*. These trees are a formalization of the analytic notation introduced by Heinrich Schenker (Cadwallader & Gagné, 1998). The basic concept is that trees are drawn in such a way that a straight branch of a tree might have another branch that attaches to the first at an angle, indicating that this second branch is subordinate to the first, straight branch. Trees start at a "root" drawn above everything else, and branches grow downwards until they terminate in concrete musical events (notes or chords). A tree so constructed makes obvious which structures are primary and which are considered elaborations of more-primary structures; this is very much like Schenker's multi-staff reductions. Trees, like grouping and rhythmic structures, are subject to a set of well-formedness and preference rules.

Surprisingly, GTTM describes two different types of reductions, each of which can be applied to the same segment of music, even though they are independent. *Time-span reductions* relate closely to the metrical and grouping structure of a piece, and also are informed by tonal cadences. *Prolongational reductions*, on the other hand, describe the tension–relaxation structure of a piece; in GTTM this is described as "the incessant breathing in and out of music in response to the juxtaposition of pitch and rhythmic factors." Time-span and prolongational reductions may interact (after all, they describe the same underlying music), and the tree structures generated may be more or less *congruent*: "Congruent passages seem relatively straightforward and square; noncongruent passages have a more complex, elastic quality."

The idea of two different reductions applying to the same music is consistent with GTTM's principle of considering the effects of not only well-formedness rules, but also multiple, possibly conflicting preference rules. Even though GTTM is a formal music theory, it maintains surprising flexibility of description. Indeed, as presented in the book it is too flexible to be used for algorithmic analysis, although several people (Hirata, Hamanaka, & Tojo, 2007; Temperley, 2001) have made computer models based on particular aspects of GTTM.

FRED LERDAHL

Lerdahl's model of tonal pitch space (Lerdahl, 2001) is not exclusively a model of melodic expectation, having perhaps more to do with harmony than melody. However, some elements of the model do produce quantitative predictions of melodic motion. In this model, the tones of the scale exist in a hierarchy of alphabets, after an idea of Diana Deutsch and John Feroe (1981). Basic tonal space, by this account, has five levels. In C-major, the levels include the following notes:

1. C
2. C, G
3. C, E, G
4. C, D, E, F, G, A, B
5. C, D \flat , D, E \flat , E, F, F \sharp , G, A \flat , A, B \flat , B

That is, the levels successively add the tonic, fifth, tonic triad, diatonic octave, and chromatic scale.

The theory provides metrics for calculating "distances" between two notes based on the number of hierarchical levels and horizontal steps between the notes in this alphabet

hierarchy. Next, formulas are provided that define distances between two chords in different tonal contexts. Later, the theory gives a way to calculate melodic tension; that is, it assigns a specific numeric tension value to each note in a melody (given the context of a key or a chord within a key). Furthermore, the theory gives a way to calculate the amount of tension in a particular chord (with respect to a given key).

Melodic tension (pitch instability) depends on the *anchoring strength* of the two pitches involved and the (squared reciprocal) distance between the pitches in semitones. Anchoring strength derives from the hierarchical level on the pitch in a space similar to the alphabet hierarchy above (in C-major, C has anchoring strength 4; E and G both have 3; D, F, A, and B have 2, and the chromatic pitches have strength 1.) The strength of attraction between two notes is calculated by dividing the anchoring strength of a possible second note by the strength of the first note and dividing by the squared distance between notes. A pitch is thus attracted to nearby pitches with high anchoring strength. This formula for attraction provides a quantitative value for the attraction between any pitch and a possible following pitch. The formula is the analogue in this theory to the expectations generated by Narmour's I-R theory (Lerdahl, 2001, p. 170)

Elizabeth Margulis provides a critique of the formula, pointing out that registral direction is ignored, semitone movement yields unreasonably high attraction values, and pitch repetition is ignored completely by the model (Margulis, 2005). Margulis incorporates an extended version of Lerdahl's model of attraction in her own model. Larson's single-level Seek Well model also includes a similar parameter of attraction – magnetism – but it, too, is augmented with other model components.

DIANA DEUTSCH AND JOHN FEROE

Deutsch and Feroe (1981) developed a cognitively-inspired model of internal pitch representation for pitch sequences. This internal representation is based on hierarchical *alphabets* of pitches and operators that refer to these alphabets to represent melodies in a compact way. Alphabets are ordered lists of pitches, such as “CDEFGAB” in the C-major scale or “CEG” in the C-major triad. Alphabets typically extend cyclically in either direction (e.g., the C-major scale can continue on to the C just above the B). Alphabets can be hierarchical, which refers to two related concepts. First, one alphabet (such as “CEG”) may be a subset of another alphabet (such as the C-major scale). Second, melodic lines may use different alphabets at different structural levels: at a higher level a melody may move through an alphabet such as the C-major triad, while at a more surface level the melody may be elaborated using notes from a superset alphabet such as the C-major or the chromatic scale.

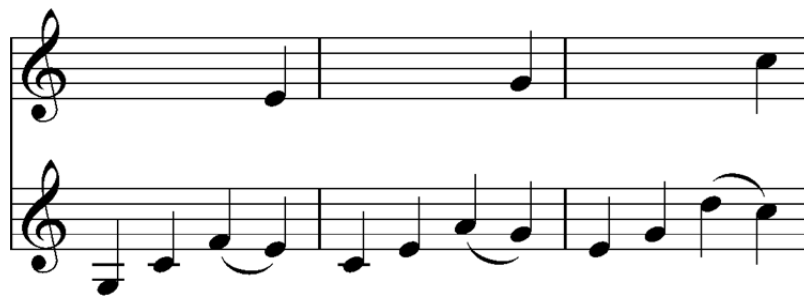


Figure 2.1: Melody described using pitch alphabets.

For example, the melody in Figure 2.1 can be heard as three sequential copies of a four-note motif that ends on a member of the C-major triad, and we can hear the higher-level structure as simply E-G-C. At the surface level, each of those notes is elaborated by a preceding upper neighbor tone from the C-major scale alphabet (indicated by slurs from F, A, and D in the figure). Incidentally, the remaining notes at the surface level are also members of the C-major triad alphabet. Given these two alphabets, we can describe the

entire melody quite compactly using simple operators such as alphabet-successor/predecessor and composition. The high-level melody line can be generated by starting with E and applying successor twice in the C-major alphabet. Then the entire melody can be generated by composing this three-note line with a four-note pattern:

predecessor of predecessor in C-major triad – predecessor in C-major triad –
 successor in C-major scale – identity

Describing melodies using alphabets and operators results in a compact representation that Deutsch and Feroe argue is cognitively plausible — they suggest that tonal music evolved to use hierarchies that take advantage of human memory structure. For tonal music in particular, different notes in a scale have different functions and degrees of stability. Hierarchical alphabets can model the relative importance of different notes in tonal music quite naturally.

DAVID TEMPERLEY

Lerdahl and Jackendoff's GTTM belongs to the world of music theory and analysis; it was cognitively inspired, but it is not specific enough to be implemented as a computer model. Temperley (Temperley, 2001), along with Daniel Sleator, implemented computer models of many components of music cognition, using principles from GTTM as inspiration. More recent work (Hirata et al., 2007) has attempted to implement GTTM itself, but for my purposes, Temperley's book (which I refer to by his acronym TCoBMS — *The Cognition of Basic Musical Structures*) is very interesting because it has goal much like my own in terms of modeling basic processes in music cognition. Of the most interest to my work is his model of melodic phrase structure, which I summarize here.

The models in TCoBMS work by optimizing an analysis of a piece of music according to a set of preference rules. For any proposed analysis, each preference rule can be used to compute a cost that measures the degree to which each musical structure generated by an analysis violates the rule. These costs are summed across all rules and structures to determine a global cost. The preference rules are all relatively local in scope and thus the global cost can be minimized efficiently using the Viterbi algorithm, a dynamic programming technique which often is applicable to music analysis. For the problem of segmenting music into melodic phrases, TCoBMS uses three preference rules, which I paraphrase here (Temperley, 2001):

1. **Gap Rule.** Phrase boundaries should be located where there are large rhythmic gaps, formed by either a phrase ending with a relatively long note or a long rest. (In the formula, actual rests are weighed twice as strongly as long note durations.)
2. **Phrase-Length Rule.** Phrases should have close to 8 notes. Phrases between 6 and 10 notes long have a low penalty, but outside of this range the cost goes up.
3. **Metrical-Parallelism Rule.** Successive groups should start at the same rhythmic position (*e.g.*, if a phrase starts on an upbeat such as beat 3 in $3/4$ time, the next phrase should also start on the same upbeat).

Temperley applied these three rules to a subset of 65 songs from the Essen folksong collection, which is notable because it is a digital collection of melodies where the data files have been annotated with phrase boundaries. (Unfortunately, hierarchical grouping structure is not indicated; if it had this feature, it would be extremely useful for studying Musicat's performance. Still, the phrase-level analysis is interesting.) Temperley's model correctly

identified 75.5% of the phrase boundaries. In a later chapter I will examine Musicat's performance on the same test set.

ELIZABETH MARGULIS

Margulis developed a model of melodic expectation that included elements of both Narmour's and Lerdahl's models (Margulis, 2005). Both tonal pitch space and innate bottom-up processing are given significant status in the model. The model provides solutions to problems that Margulis had pointed out with each in her critiques. For example, it explicitly describes how expectation connects to the experiences of affect and tension, how to deal with repeated notes, and how hierarchy can be used formally to include more than the two preceding notes in the generation of expectations. The model is composed of five separate components: stability, proximity, direction, mobility, and hierarchy.

Model Components

Stability

Stability of melodic events is calculated based on the tonal context. Rules adapted from Lerdahl (2001) select a chord and current key for each pitch event. Based on the tonal function of each pitch, a stability rating is assigned. These are numerically similar to the anchoring strengths in Lerdahl's theory. However, they are more sophisticated because they are based on the current tonal context. For example, several exceptional cases such as augmented sixths and Neapolitan chords are enumerated in the model to improve the quality of stability predictions.

Proximity

Just as in Narmour's I-R model, the principle of proximity states that listeners have higher expectations for pitches nearby in frequency. This model gives a numerical proximity rating to pitches based on the distance in semitones away from the preceding pitch. Margulis selected the particular numerical values by hand based on results from various studies and on her own intuitions.

Direction

Narmour's I-R model states that small intervals imply continuation of direction but large intervals imply reversal. Margulis incorporates this idea to generate particular expectations (for continuation or reversal) along with the strengths of each expectation. These expectations are based on the interval size measured in half-steps. Instead of prediction strength being a simple linear function of interval size, Margulis uses data from Schellenberg's study on simplifying the I-R model (1997) to suggest a particular nonlinear mapping from interval size to prediction strength.

Mobility

Although many theories ignore the possibility of repeated notes, Margulis includes a mobility parameter that increases the degree to which a note is expected to move to different pitch. Repeated notes lead to a mobility penalty whereby the stability and proximity scores are multiplied by $2/3$ to encourage motion. Margulis notes that this parameter was explicitly added to reduce the strong expectations for repetition otherwise produced by the model.

Hierarchy

Without the concept of hierarchy, the amount of expectation for a pitch to follow another pitch is given by the model as **stability × proximity × mobility + direction**. In

other words, it's good if the second note is close to the first one in pitch, if the second note is a stable pitch, and if the motion is in the expected direction. Hierarchy is incorporated in the model by generating a time-span reduction of the input and then applying the formula above to each level of the hierarchy. The final expectancy value for each pitch (expectancy is simply the degree to which a pitch is expected) is given by a weighted average of the values at each level. Margulis selected the weights in this formula by hand; the surface level has a weight of 15, other levels of no more than two-second duration have weights of 5, remaining levels with less than a six-second duration have a weight of 2, and no levels are considered with a time span longer than six seconds. In the model, the time-span reduction is generated via an implementation of preference rules from GTTM (Lerdahl & Jackendoff, 1983), augmented with an additional preference rule.

Tension and Affect

This model defines three different types of tension based on calculated expectancy values: surprise-tension, denial-tension, and expectancy-tension. The first two types are calculated for a third event in a series based on the two prior events. For example, the notes A-B in a C-major context may set up an expectation for continued upward motion to the tonic C, but if the pattern continues A-B-F# the third event yields a high value for denial-tension. Expectancy tension, on the other hand, is calculated when looking forward to a potential expected future event. Thus, in our example, the second event, B, may have a high value for expectancy tension if the tonic C is strongly expected. Expectancy tension describes the strength of a future expectation, whereas the other two types describe the amount of surprise or frustration that result from an event that has just occurred.

Surprise-tension describes how unexpected an event was. In other words, its value is inversely proportional to the expectedness of the event. An event that was not expected at all results in high surprise-tension, while an event that was expected to a moderately high degree results in moderately low surprise-tension. In a C-major context, a sudden, unprepared F# would result in high surprise-tension. Margulis describes high values of surprise-tension as being associated with an experience of “intensity” and “dynamism”.

Denial-tension is proportional to the difference between the expectancy value for a note that occurred at a particular time and the maximum expectancy value over all possible notes that could have occurred at that time. Thus denial-tension is strong when there is a very strongly expected note that does not occur, or when the actual note is quite unexpected relative all possible note options. For example, consider a melody moving up the C-major scale from C to the leading tone, B. If the melody continued by moving down to A instead of moving up to the tonic, C, it would result in high denial-tension, because the tonic is so expected after the leading tone. Margulis writes that high denial-tension results in feelings of “will, intention, and determinedness”.

Expectancy-tension is proportional to the strength of the most-expected following note. Its calculation is thus quite similar to that of denial tension except that it is forward-looking. In the example of the C-major scale above, there would be a large amount of expectancy-tension at the moment the scale reached the leading-tone, B, because the tonic C is so much more expected than any other note. Expectancy-tension is associated with feelings of “yearning” and “strain”.

Margulis uses the three tension formulas to generate graphs of each type of tension versus time for musical pieces.

DAVID HURON

Huron's book *Sweet Anticipation: Music and the Psychology of Expectation* (2006) gives a broad overview of research relevant to musical expectation. In the book Huron develops his "ITPRA" theory of expectation. Like Margulis's model of expectation, Huron's model describes emotional states that come along with the experience of expectation. ITPRA stands for *imagination, tension, prediction, reaction, and appraisal* — all of these are stages in the process of expectation. Notice that ITPRA is a general model, applicable to other domains than music.

Huron's model is fascinating, but expectation is not completely central for Musicat in its current state. Therefore, I will not describe the model in detail here. However, I will point out a few key sections of the book that I found particularly relevant.

Heuristic Listening

In Huron's book, the title of Chapter 6, "Heuristic Listening", is especially interesting to me because I think of Musicat as a heuristic listener. A key element in this chapter is a comparison of statistical features of Western melodies with experimental results of how people actually listen. Huron gives four statistically common musical patterns, but three of the four seem to be represented by imperfect heuristics in listeners' expectations:

1. Pitch proximity (the tendency for the next note in a melody to be close to the previous one) is found both in statistical analysis of melody and in listeners' expectations for the next pitch.

2. Regression to the mean (melodies tend to revert back toward the mean pitch) is approximated by listeners expecting post-skip reversal (a large leap should be followed by a change in pitch direction).
3. Downward steps are the most common melodic motion, but listeners instead expect inertial motion for small intervals (in either direction) to continue in the same direction.
4. Arch phrases are common (melodies tend to begin by ascending and to end by descending) but listeners only expect the second half – descending pitches in the last half of a phrase.

Tonality

Scale Degree Qualia

Huron performed a short survey of listeners' experiences in Western tonal music, asking them to describe the individual qualities of each chromatic scale degree in a musical key (p. 144–147). The survey resulted in quite rich descriptions for each note, and suggest that a model of musical listening needs to somehow incorporate this sort of knowledge. Some responses were unsurprising: the tonic was described as “stable” or “home”, but also as being associated with “pleasure” or “contentment”. The mediant was described using words such as “bright” and “warmth”, but also with the much more emotional-laden words “beauty” and “love”. The dominant tone was “strong”, “pleasant”, and even “muscular”. Huron found many types of scale degree qualia that were shared across listeners, and speculates on their origins. These qualia seem crucial to the human experience of listening, but current models (Musicat included) barely incorporate this sort of knowledge at all. This

is clearly a vast and complex aspect of musical listening that is important for future work in music cognition.

Cadence and Closure

Huron discusses how points of temporary closure in music are related to tonal pitch patterns (pp. 143–157). The notion of “cadence” is well-explored in music theory: a cadence is a specific pattern of melodic and harmonic elements that signify closure in Western tonal music. Different cadences can signify different amounts of closure. Huron notes that even non-Western music has cadential patterns, and within tonal music, different composers and different genres have different types of cadence.

Huron refers to the discipline of “information theory” to describe cadential patterns in terms of the statistical predictability of certain patterns of notes. According to the Shannon-Weaver equation, the notes that occur just before a point of closure (such as a cadence) have low uncertainty (high predictability). A cadence is often heard as a “reset” point, and notes following a cadence have high uncertainty (they are hard to predict). Huron mentions that Narmour’s conception of closure can be summarized in a way that supports this view of cadence as a “reset point”: Margulis has said that for Narmour, closure is “an event that suppresses expectancy” (p. 157).

In summary, cadences are learned patterns in tonal music that establish a feeling of closure. Although we can understand parts of music cognition in terms of gestalt theory, cadential patterns must be learned for each particular musical tradition.

Expectation in Time

Huron describes a set of experiments by Caroline Palmer and Carol Krumhansl to study listeners' perceptions of musical meter and metric hierarchy, inspired directly by Krumhansl and Edward Kessler's probe-tone experiments in perception of tonality. Interestingly, Huron makes a very strong association between tonal hierarchy and metric hierarchy:

But the similarity between scale-degree expectation and metric expectation is not merely metaphorical or informal. Both scale degree and metric position are perceived categorically. Like scale-degree pitches, metric positions provide convenient "bins" for expected stimuli. The metric hierarchy is truly homologous to scale or scale hierarchy. (p. 179)

This analogy between pitch and meter was quite unexpected to me, and I find it quite provocative and exciting. When a melody ascends through a major scale from the tonic up to the tonic an octave higher, it starts on a very stable pitch (say, C, in C-major), moves past a unstable pitch (D), then a somewhat stable pitch (E), a less stable pitch (F), a very strong and rather stable pitch, the dominant (G), a less-stable pitch (A), an extremely unstable pitch, the leading tone (B), and finally it arrives on the tonic, again, a very stable pitch. The pattern of stability alternates quite regularly between stable and unstable (aside from the leading tone), and even the degree of stability fluctuates in a regular pattern: C and G are more stable than E, sandwiched between them and G itself is less stable than the two tonic C's that surround it. This pattern of alternation is quite reminiscent of the pattern of alternating strong and weak beats we find in a metric hierarchy. Both patterns look something like the markings on the edge of a ruler that denote inches, half inches, quarter inches, and so on. If only our default time signature were 7/4, the analogy between pitch and meter would be quite elegant! Still, the analogy suggests that the same (or similar) cognitive mechanisms might be applicable in both the pitch and meter domains.

Binary Default

Huron describes some research (p. 195) supporting the idea that Western listeners expect “binary” structures in musical meter. Even without resorting to experimental data with human subjects, a survey of Western classical music suggests that meters based on two or four beats are much more common than other meters. Specifically, Huron used over 8000 melodies from Barlow and Morgenstern’s *Dictionary of Musical Themes*, and discovered that 66% of them used a binary meter of two or four beats per bar. (Incidentally, in a study I conducted, 79% of participants’ improvised melodies were in duple or quadruple meter — see Appendix B.) Huron’s results are not surprising, but they are important because it helps to justify building in a strong duple-meter bias in computer models such as Musicat.

BOB SNYDER

Music and Memory (Snyder, 2000) details how human memory processes influence music cognition in many different ways, across time scales ranging from a few milliseconds to many years. Unlike many other references in this chapter, this book’s aim is to synthesize existing work in memory-related aspects of music cognition, rather than present a new theory or computer model; as a result, it examines a surprisingly wide range of complex musical phenomena and cognitive processes. Some highlights relevant to my work include the book’s overview of memory processes, closure, melodic schemas, categorical versus parametric aspects of music, rhythmic tension, and hierarchies of musical chunks. (Other sections of the book such as those on gestalt grouping and GTTM have been addressed earlier in this chapter.)

Memory Processes

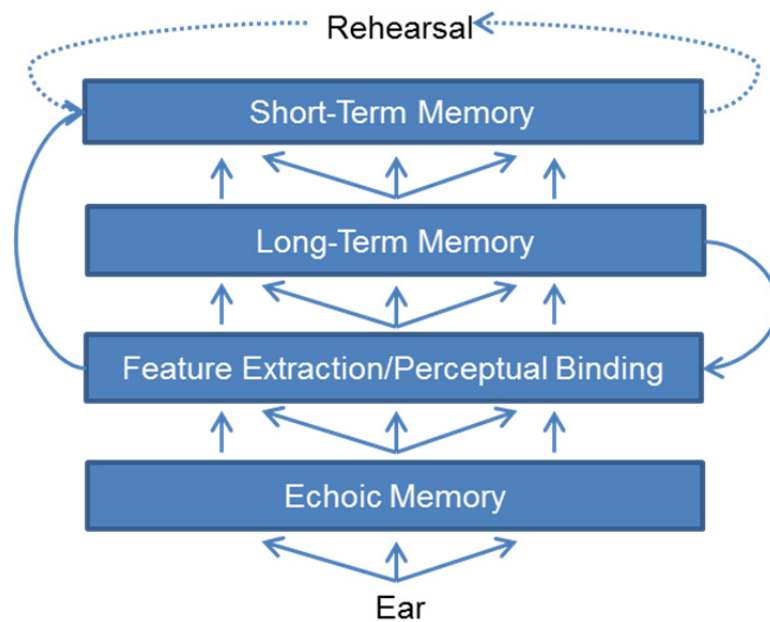


Figure 2.2: Auditory memory processes, adapted from (Snyder 2000).

Music and Memory begins with a chapter describing several important cognitive processes involved with music perception. Three different types of memory appear in a figure on p. 6, shown here in quite simplified form in Figure 2. Sound comes into the ear and is separated into its component frequencies by the clever architecture of the inner ear and some neural circuitry. This information persists for a short time in the *echoic* memory, a sort of short-term buffer where we can remember the past few seconds of raw perception. Traces of sound “echo” here long enough for slightly higher-level perceptual features such as individual notes, percussive events, and timbres to be extracted. These perceptual features can interact with and activate categorical structures in long-term memory (LTM), resulting in perception of music as sequences of larger conceptual structures (such as musical motifs). The stream of low-level perceptual information can also be sent directly to short-term memory and conscious awareness, where it will be perceived in a fleeting way that Snyder describes as

“nuance”. Nuance refers to the details of an instance of a perceptual category that differentiate the instance from the generalized category.

Most of the processes in the diagram are unconscious. Snyder describes a window of conscious awareness in which the structures in short-term memory are available for (completely) conscious examination. The process of rehearsal (the conscious and internal repetition of a fragment of music) helps make it more likely that the current perceived structures will get stored in LTM. This is important because only the most recent three to five seconds of music in short-term memory (STM) are generally available for conscious scrutiny.

The time scales involved with both memory-based processes and perceptual processes interacts with the time scale of musical structures in an interesting way. Snyder divides musical structures into three main groups based on the time scale involved: event fusion (hearing sound waves as musical events), melodic and rhythmic grouping, and form. These correspond to three memory-based or perceptual processes: early processing, STM, and LTM. Early processing occurs for events that are less than approximately 1/20 second long (frequencies above 20 Hz), STM is most involved in processing for events lasting up to about 8 seconds (such as melodic grouping and phrasing), and anything longer than that (longer musical sections, entire movements or works) must have more to do with LTM.

Categorical vs. Parametric Aspects

In a chapter on large-scale form, Snyder is concerned with how boundaries are perceived between sections. He asserts that large-scale boundaries are points of “multiparametric change”, where each section internally has “parametric constancy” (of course, these sections can be of various time-scales, so change and constancy are relative

terms that take into consideration the time-span in question.) But what are these various parameters which may be perceived as changing at section boundaries? Snyder divides them into *primary* and *secondary* parameters, referencing Meyer and Hopkins. This division is helpful in thinking about modeling various aspects of music.

Primary parameters are those that can be perceived categorically (*i.e.*, parameter values belong to discrete categories), such as notes in a scale or time segments made up of fixed rhythmic units. Microtonal variation in a note (such as vibrato) would not be primary, but hearing a note as having an identity such as “the fifth degree of the major scale” would be primary. Snyder identifies three primary parameters: pitch, harmony, and rhythm.

Secondary parameters are those that vary continuously, such as volume, tempo, or timbre. Even though we very clearly perceive changes in these parameters, they cannot be represented in memory in as rigid a way as the primary parameters.

Tonality and Melodic Schemas

How is the pitch content of a melody organized in memory? In addition to possible *local* representations such as absolute pitch, intervals between pitches, and pitches as elements of an “alphabet”, the *larger-scale* structure of pitch in a melody is important. Snyder (like Schenker and others) discusses tonality and melodic shape as organizational forces, but in Snyder’s book these are discussed in terms of human memory constraints.

Memory is particularly important for tonal organization because tonality is only perceived in context as musical time moves forward. Some music-informatics approaches to tonality, in contrast, consider collections of pitches as an unordered set (especially those that make use of the Krumhansl–Kessler key profiles (Carol L. Krumhansl, 1990) for key estimation). However, the order of pitches in a melody matters greatly in perceiving tonality (Butler & Brown, 1994). According to Snyder, part of establishing tonality is that the

repetition of a pitch causes it to become central. Other musical parameters also help emphasize key pitches: notes occurring on strong metric positions or with long durations are emphasized. Once tonality is established, return to the central pitch can help establish a sense of closure (see below). Tonality establishes expectations and specific goals, and in turn, reaching these goals aids in establishing grouping boundaries.

In addition to perceiving individual pitches in terms of tonal structure, people hear sequences of pitches in terms of melodic schemas. The most well-known such schema is perhaps the *melodic arch* form: many melodic structures gradually rise to a high note and then fall back down rapidly to the starting pitch. The tendency for the climax to be found closer to the end than the middle of the segment is more pronounced for larger-scale structures. (See the previous section on Huron for more on melodic arch and expectation.) Another melodic schema is the *axial melody*, in which a melody will fluctuate about a central pitch — good examples are found in Gregorian chant. Whereas the melodic arch involves rising from and falling back to a stable base (as in Larson’s notion of musical gravity), an axial melody significantly involves notes both above and below the central (not base) pitch. A final type is the *gap-fill* schema, in which a melodic leap forms a “gap” and the melody eventually moves in the opposite direction, to “fill in” the missing notes in the gap — the vertical gap between pitches is heard as forming a melodic “vacuum” and the melody notes are “sucked” into the space left between the two pitches involved in the leap.

Melodic schemas offer one way in which we can understand a melody as it progresses in time (as they help suggest goals and expectations). After a melody has reached closure, we may internally represent these larger-scale melodic shapes as belonging to one of these schema types (*i.e.*, the schemas represent broad categories of stereotypical melodic shapes, and we may hear melodies as members of these categories).

Closure

Snyder invents the term “soft closure” to describe the weakest kind of separation in music that causes a grouping boundary. For example, a melodic leap or a relatively long time interval between notes could create soft closure and a very local grouping boundary. He makes a rather bold claim about how the number of parameters establishing closure determines the hierarchal importance of a boundary:

Melodic or rhythmic grouping boundaries are usually established by changes in one basic parameter, such as pitch contour or duration, whereas phrase boundaries are usually reinforced by changes in *more than one* parameter...
(p. 38)

I find this difference between just one and more-than-one parameter to be too simplistic-sounding. However, Snyder softens the statement by going on to mention that “the distinction between grouping and phrase is not absolute”. Later, he makes the more general statement that higher levels of structure attain closure by the simultaneous closure of a larger number of musical parameters. The most definite closures are established through culturally-specific musical syntax, such as cadential patterns, but even without the syntax of a particular musical style, changes in multiple parameters can establish a strong sense of closure.

Closure is also strongly tied to expectation. If a musical pattern does not suggest particular expectations for the listener, and then it ends, it will not impart a feeling of closure; it will just sound like starting followed by stopping. Tension and expectation are required before closure can happen. In addition to the things mentioned above, such as cadence and multi-parameter change, which may imply closure, closure can also result from repetition of previously-heard musical material. For example, the ABA form (which can function at many different levels of hierarchy) relies on the contrasting B section to establish some tension, and then the repetition of the initial A section releases the tension and leads to a feeling of closure.

Rhythm

In order to be perceived as such, a rhythm must take place on a short enough time scale that it may be represented in short-term memory — a maximum of five seconds or so. (p. 162) According to Snyder, this is necessary because a rhythm is defined in terms of relative time intervals, and the rhythm needs to be held in memory all at the same time so that the relationships between time intervals can be interpreted correctly. Also, rhythmic perception is categorical — that is, we hear elements of rhythms as belonging to discrete categories such as quarter-note and half-note, not in terms of continuous parameters such as “number of seconds of duration” — as is shown, for example, by the way in which humans can recognize the underlying rhythmic structure of a *rubato* passage despite the distortions it involves). The subtle shifts in tempo and attack times and durations that make for expressive performance are important for listeners, but rhythms still retains their identities during this kind of performance. It seems, then, that short rhythms are encoded in memory in a categorical manner where time intervals are heard in terms of small-integer ratios relative to each other or to a basic, lowest-common-denominator pulse.

It is natural to talk about musical tension in the pitch domain (think of the importance of the highly-tense tritone), but Snyder also defines several useful terms for thinking about tension in rhythm and meter. *Metrical tension* refers to the difference between the accents expected in a particular meter and the accent structure and rhythmic grouping of the actual music. For example, hearing the start of a group on a weak beat adds to metrical tension. *Rhythmic tension*, on the other hand, is unrelated to the meter; it refers to the internal tension created by rhythms using shorter-duration notes and causing a perception of “speeding up” (even when tempo remains constant). For example, a melody in quarter notes that suddenly incorporates several successive sixteenth notes will be heard as having greater

rhythmic tension. *Rhythmic contour* refers to the pattern of fast versus slow elements inside a rhythm, and we can imagine a graph displaying rhythmic contour as the rising and falling of note speed (or equivalently, note density) over time.

Hierarchies of Chunks

As is well known in cognitive science, working memory capacity is limited, roughly by the famous “magical number 7 ± 2 ” study (G. Miller, 1956), and so we use *chunking* to organize information into units that can be held in memory as single objects, thus using up much less working-memory “space”. Chunks can form hierarchies so that we can still reason about complex structures, “unpacking” individual chunks or sub-chunks as is necessary when the details are relevant. Naturally, music is heard in terms of these hierarchies of chunks. This isn’t to say that an entire piece is represented for a listener as a perfect hierarchical structure: “It is not clear, however, how many levels of hierarchical organization are obvious to musical listeners” (p. 218). We are able to perceive at least certain local portions of the musical hierarchy, and to hear the large-scale structure of a piece as several large sections. Chunks of music are stored in memory and we can most easily recall music from memory by remembering events at the chunk boundaries. This phenomenon naturally is not specific to musical memory, but rather is a general consequence of the nature of human memory and occurs with other types of time sequences in memory.

Snyder gives an example of the way in which children typically learn to speak the English alphabet in a particular rhythmic manner, which suggests the following chunk-based organization (p. 219):

ABCD EFG HIJK LMNOP QRS TUV WXY & Z

Note that each chunk of letters has no more than five letters, less than the “magical number seven”. Snyder suggests that as there are only seven chunks here, the entire alphabet may be represented simply by this sequence of seven small chunks. I suspect there is more structure, however, especially since the rhythm is derived from the “Twinkle, Twinkle, Little Star” melody, which has additional hierarchical structure — see later chapters for more on that melody. For example, “QRS” and “TUV” have the same rhythm (whether spoken or sung) and probably form a mid-sized chunk made up of these two pieces: “QRS TUV”. The following representation shows the additional structure.

$$\left\{ \left[(ABCD\ EFG)\ (HIJK\ LMNOP) \right] \left[(QRS\ TUV)\ (WXY\ \&\ Z) \right] \right\}$$

STEVE LARSON

Three Musical Forces

Steve Larson and FARG developed the creative microdomain Seek Well² to study melodic expectation (Larson, 1993b). To reduce the potentially overwhelming complexity of the domain while retaining key features of interest, Seek Well involves melodies in the classical tradition (*i.e.*, Western tonal music) conforming to the following list of restrictions:

- Only one note sounds at once; the sound represented is monophonic.
- All notes have the same duration.

² To avoid a possible source of confusion, I should point out that Steve Larson used the term “Seek Well” in two different contexts. As I used the term here, it is the name of a particular microdomain. It can also be used to describe the “Seek Well” project, which refers to a set of computer programs that work in this microdomain (*i.e.*, the “Single-level” and “Multi-level” models described below).

- All notes sound the same except for pitch (dynamics, articulation, etc. are not involved).
- Rests are not allowed (notes come immediately one after another until the melody ends).



Figure 2.3: Beginnings of typical Seek Well melodies.

Figure 2.3a is a typical melodic beginning in the Seek Well domain. Larson studied typical responses to this cue by asking listeners to sing a completion of any length (Larson 1997). Two common continuations are shown in Figures Figure 2.3b and Figure 2.3c. Note how the response Figure 2.3b implies the listener heard the cue in C-major, while response Figure 2.3c implies F.

Larson's theory of musical forces states that "we tend to hear music as purposeful action within a dynamical field of musical forces" (Larson, 1993a), making an analogy between physical motion through space and the perceived "motion" of a melodic line. The three forces involved are musical gravity, magnetism, and inertia.

Gravity refers to a tendency of notes heard above a stable platform to descend to that platform. For example, given the rising line C-D-E in C-major (Figure 2.4a), gravity would suggest a continuation back down to the stable tonic: C-D-E-D-C (Figure 2.4b). After the

alternate beginning C-D-C (Figure 2.4c), however, gravity does not continue to pull the line down past C, because C has been established as a stable base.



Figure 2.4: Three musical forces.

Magnetism refers to the perceived attraction between notes of unequal stability. For example, in the ascending octave that stops on the leading tone, C-D-E-F-G-A-B (Figure 2.5a), we feel the strong magnetic force of the stable upper octave C “pulling” on the unstable B, leading to the expected completion in Figure 2.5b. As with physical magnetism, the strength of the force is inversely proportional to the distance between two pitches. The B would also be attracted downward to the stable G, but the magnetic pull of the upper C is stronger because it is three times closer, as measured in semitones.



Figure 2.5: Ascending octave.

Inertia is the tendency of a musical pattern to continue “in the same way”. A simple example is the tendency of a moving line to continue moving in the same direction. Given the same beginning as in the gravity example, C-D-E (Figure 2.4a), inertia suggests that instead of falling back down to C, the melody might continue rising through F, G, A, etc., as in Figure 2.5. Musical inertia corresponds to the physical statement “Bodies in motion tend to stay in motion”. Another example of inertia is an Alberti bass, which tends to continue its characteristic pattern once set in motion.

These three forces act continuously on musical lines in a dynamically shifting musical context. A significant part of this context is provided by pairs of *reference alphabets* and *goal alphabets* (Larson, 2004), inspired by Deutsch & Feroe’s (1981) notion of pitch alphabets described above. *Reference alphabets* are tonal pitch sets through which a melody moves, such as the C-major scale in the examples above. *Goal alphabets* are subsets of reference alphabets that serve as stable, goal points for melodic motion, such as the members of the tonic triad in these examples. Thus we can say that musical lines move *through* a reference alphabet *between* notes of a goal alphabet. For instance, the common descending melodic pattern 5-4-3-2-1 in major moves through the major scale reference alphabet (composed of all seven pitch classes of the scale), starting and stopping on the tones of the simple tonic–dominant goal alphabet (consisting simply of 1 and 5).

It is illustrative to reconsider the ascending octave line (Figure 2.5b) in the context of all three forces. The melody starts out at rest on C, the stable base. An external force starts the motion by “pushing” the melody up to D. The external force subsides and the melody begins to be tugged in various directions. Gravity exerts a constant force pulling downwards, back towards C. Magnetic forces are pulling with equal strength up to the E and back down to C. The inertia of the initial push keeps the melody going upwards to E, overcoming the force of gravity. At this point the magnetic pull of G becomes prominent as it is closer than

the lower C.³ Magnetism accelerates the melody up through F towards G, and once it has reached G, it sails past it because of inertia. However, the downward tug of gravity is still noticeable, as is the strong magnetic pull of G, once the melody has ascended to A. Only inertia enables the melody to rise a bit further to B. Once it has reached B, the magnetic force of the upper C is irresistible and overcomes both gravity and the magnetism of G, pulling the melody all the way to C.

The dynamic push and pull of the forces is significant in that it acknowledges some of the complexity of how we listen to music. However, the preceding description may sound overly mechanical — surely the path taken by a melody does not *really* act like an physical object moving through physical fields of force. After all, if this were the case, then an initial C-D melody would always result in the melody traversing the whole octave! Larson's theory avoids this absurd scenario because he claims that many of these descriptions result from *retrospective* hearing (Larson 2004, personal communication). The melody above might well have stopped short of reaching the upper-octave C, but in that case we would have come up with an alternate story describing which forces prevailed at each point in the melody's path. Because the forces interact in a flexible manner, they can provide explanations for alternate melodic continuations. For example, we might imagine that the initiating "shove" was not strong enough to build up inertia to carry the melody past F, A, or even the initial D (Figure 2.6a,b,c below). In each of these cases, the initial inertia-based ascent "ran out of steam", as it was overcome by gravity or magnetism or both. Musical forces can also explain more

³ In Larson's theory, magnetism is based solely on the distance from a note to the nearest "stable" note. The tonic and dominant scale degrees each induce the same base amount of force. We can imagine an extension to the theory, based on an analogy with physical magnetism, in which the tonic would exert a larger amount of magnetic force than the dominant, as measured at a point equidistant from both. In this system, the attractive force from the leading tone up to the tonic would be much greater than the force pulling the raised fourth degree to the dominant.

extended motion such as that of Figure 2.6d: the initial ascent to A is followed by a descent via gravity and magnetism back down through G, a brief lift back up to F due to the magnetic pull of G, and a final giving-in to gravity to return to the stable platform of C (after briefly dipping below the base due to inertia).



from a predefined list. Several rules help ensure that the alphabets chosen make musical sense in the context of the force under consideration. For instance, when the model is making a gravity or magnetism prediction, the cue must end on a pitch that is present in the reference alphabet but not in the goal alphabet (that is, the final note of the cue must be heard as unstable so that there is an impetus for continuation). Next, the model examines just the final three notes of the cue and produces predictions based on these notes, the musical forces acting on the notes, and the pairs of alphabets under consideration. These predictions involve motion through the reference alphabet to a stable member of the goal alphabet, providing a sense of completion. The predictions are assigned probabilities according to the strength of the forces as defined by the theory. Thus, the output from the model is a set of different predictions and associated probabilities. Finally, if desired, the resulting predictions can be fed back into the system as new input so that it can predict what will follow each of those possible continuations, generating longer predictions.

Multi-Level Model

The theory of musical forces applies not only to the surface-level notes in a melody, but also to deeper levels of an embellishment hierarchy⁴, which is a simplified Schenkerian analysis that can be described in a strict form amenable to computer representation. In Figure 2.7 there are two levels in the hierarchy. Staff A represents the deepest (background) level, while staff B depicts the foreground level, made up of the notes actually heard by the listener. While there are only two levels of hierarchy in this example, there could be several middleground levels in more extended examples. Any discussion in this paper of relations

⁴ “Embellishment” is a technical term from Schenkerian analysis. In brief, notes at the musical surface are said to embellish notes that are part of a deeper structure. Larson argues that embellishment is equivalent to the notion of “prolongation” in Schenkerian analysis — see (Larson, 1997b) for discussion.

between the background level and the surface applies equally well when the background or surface is replaced by a middle level.



Figure 2.7: A simple melody (below) and an implied background level (above).

The current implementation of the multi-level model requires the user to provide not only the surface notes of the melodic beginning and the key of the melody, but also a specific type of Schenkerian analysis. In this case, both staves A and B would be supplied to the computer, along with a description of the embellishment function of those notes not present in the background level. For instance, the user might describe the initial note B as a “prefix chromatic neighbor” to the following C. This type of analysis differs from a typical Schenkerian analysis in its specificity: the precise embellishment function is explicitly described in the analysis. At the same time, the theory also claims that the analysis must be flexible in describing the embellishment function. For instance, in some musical contexts a description such as “suffix diatonic lower neighbor” might be appropriate, while in other cases the description might be generalized to “suffix diatonic neighbor”.

The multi-level model considers each level as an individual melody line and generates a completion of each line. In this example, the *multi*-level model would begin a prediction by applying the *single*-level model to staff A. The single-level model would generate middle C as a likely final note, due to the influence of both inertia and gravity. (The other possibility, discussed below, would be a return to G based on the magnetism between E and G.) Here the reference alphabet in use is the C-major triad, with a goal alphabet made up of only the

tonic and dominant. Figure 2.8 demonstrates the partially completed prediction after this step.



Figure 2.8: Prediction at a higher level.

Having generated a completion for the background level, the multi-level model proceeds to complete the surface level. This step depends on the particular analysis supplied to the system. In the current example, if all the embellishment notes are described as “chromatic lower-neighbor prefixes”, the force of inertia indicates that to continue “in the same way” the final C should also have this type of prefix embellishment. The word “chromatic” indicates that the alphabet to be used for the prefix note is the chromatic scale (as opposed to the triadic alphabet used in the background level). Hence, the predicted embellishment note is the note B as shown in the analysis in Figure 2.9. In this diagram, the slurs connect embellishment notes (shown without stems) with notes present at a deeper level in the analysis.



Figure 2.9: Surface-level prediction.

Future Models of Musical Forces

The single- and multi-level models implemented by Larson are only a first approximation to his complete theory, which is described more fully in his book (Larson, 2012). The Seek Well subsection later in this chapter under the FARG heading discusses how the theory of musical forces can be implemented using the ideas in other FARG computer models.

Musical Rhythm

Rhythm plays a particularly primal role in human listening, as well as in the Musicat program. The following sections describe a few pieces of relevant research focused on musical rhythm and cognition. I emphasize that the three sections below just scratch the surface of work on this area. In particular, two important books not described in detail here might be useful in future work on the Musicat project: *The Rhythmic Structure of Music*, by Grosvenor Cooper and Leonard Meyer (1960), and *The Stratification of Musical Rhythm*, by Maury Yeston (Yeston, 1976). Both of these books discuss rhythms with respect to multiple levels of hierarchy, which fits in well with the Musicat's generation of hierarchical grouping structure. Both books discuss relationships between pitch and rhythm — an issue critical to further development of Musicat — and Cooper and Meyer, specifically, discuss many subtleties of rhythm that seem relevant to Musicat.

I suspect that a careful reading of these Cooper and Meyer as well as Yeston would inspire more heuristics and design ideas for future versions of Musicat. The work described below, however, has already proved useful in the development of Musicat.

DIRK-JAN POVEL AND PETER ESSENS

Povel and Essens (1985) designed a model of meter perception in a pitch-free domain. The central idea motivating the model is that music causes a sort of ticking “hierarchical internal clock” to be induced in the mind of a listener. Given a series of time points representing note attacks, the model selects the most likely metric structure from a collection of simple meters involving a two-level hierarchy of internal clocks. For example, meters such as 2/4, 3/4, 4/4, and 6/8 can be heard, and these are distinguished based on how accented beats are arranged in a two-level rhythmic hierarchy (*e.g.*, 3/4, a simple triple meter, has a default accent pattern based on a single level with three beats, whereas 6/8 has a different accent structure because it is a compound meter made up of a higher two-beat level combined with a lower three-beat level). Even though there are no differences in the volume or strength of each attack point in the input to the system, it uses heuristics to detect attack points which are perceived to be louder (for instance, attacks at the start or end of a group are perceived to be stronger than those in the middle of a group as a result of perceptual primacy and recency effects). The model selects the best clock simply by comparing each possible clock from a limited set of clocks to the pattern of attacks in the input; the clock that scores the best according to a metric based on shared accents and minimized missing accents is selected as the winner.

Povel and Essens hypothesized that listeners will be able to reproduce a rhythm more accurately when it is amenable to encoding in a simple way with respect to the stresses of the induced internal clock, and verified that this was indeed the case using experiments with human subjects.

EMILIOS CAMBOUROPOULOS

While Povel and Essens give an algorithm for determining musical meter, Combouropoulos (1997) presents a computer model (the Local Boundary Detection Model, or LBDM) that infers, based on local melodic structure, a set of *potential* boundary points. The points discovered by the model might form the starting or ending points of higher-level musical groups, but the output of the model is a list of candidate group boundaries with associated degrees of confidence. Especially relevant to my work is how the model is intended to be used in conjunction with a model involving higher-level structures and relationships such as parallelism:

“...one has to bear in mind that musically interesting groups can be defined only in conjunction with higher-level grouping analysis (parallelism, symmetry, etc.). Low-level grouping boundaries may be coupled with higher-level theories so as to produce *optimum* segmentations.” (p. 279)

LBDM uses two simple rules, based on gestalt proximity, for suggesting boundary points: the Identity-Change Rule (ICR) and the Proximity Rule (PR). ICR considers a set of three successive notes and examines the two parametric intervals formed (the interval between the first two notes and the interval between the second and third notes). Note that these may be time intervals (*e.g.*, durations such as “1 beat” or “2.5 beats”), but the rules are not restricted to the domain of time — they apply to pitch, dynamics, articulations, etc. If the two intervals are different, the rule suggests a boundary point located inside each interval.



Figure 2.10: No boundaries.



Figure 2.11: Two boundaries.

For example, considering the rules applied to note start times and given the sequence Quarter-Quarter-Half (Figure 2.10), no boundaries are implied because both time intervals

are one quarter-note beat long. However, for the sequence Quarter-Half-Quarter (Figure 2.11), the first interval is one beat, while the second interval spans two beats. Two boundary points are indicated; one between beats 1 and 2, and one between beats 3 and 4. Now, the other rule comes into play to adjust the relative strengths of the boundaries: PR gives a stronger preference to the larger interval. In this case, the boundary after the half note (between beats 3 and 4) is stronger. The LBDM model is applied to several different musical parameters and boundary point strengths are summed across these parameters, yielding a strength value for each possible boundary point.

Combouropoulos gives two theoretical applications of LBDM in addition to its use in discovering grouping structure. First, he suggests that the “phenomenal accentuation structure” — that is, the structure of the *perceived* accents in music — can be derived from the model: local maxima in the boundary strengths should correspond to phenomenal (perceived) accents. Note that again here Combouropoulos is careful to point out that there may be ambiguity in the analysis, which should be resolved by using the model in conjunction with other higher-level considerations. Second, given this accentuation structure, various well-formed metrical grids may be superimposed on the structure to find a best match; the best match is a candidate for the true metrical structure of the music.

WILLIAM ROTHSTEIN

In *Phrase Rhythm in Tonal Music*, Rothstein (1989) introduces the concepts of phrase, period, meter, hypermeter, and phrase expansion from a contemporary music theorist’s perspective. Several ideas from this discussion are particularly relevant here.

To answer the question “What is a phrase?” Rothstein quotes two theorists: Roger Sessions and Peter Westergaard. Sessions writes that “The phrase is a constant movement

toward a goal — the cadence.” According to Rothstein, Sessions’ concept of the cadence was primarily rhythmic. On the other hand, Westergaard’s definition is based on tonal motion. A phrase has these properties:

1. establishes one set of pitches and then
2. moves to a second set of pitches in such a way that
 - a. we expect those pitches
 - b. we have some sense of when they are about to occur, and
 - c. once they have occurred, we know the phrase has gotten where it’s going and that no further pitches are needed to complete that phrase.

(Rothstein, 1989, p. 4; Westergaard, 1975, p. 311)

Rothstein’s own view (p. 5) is that “a phrase should be understood as, among other things, a directed motion in time from one tonal entity to another... *If there is no tonal motion, there is no phrase.*” This is a useful heuristic for ruling out certain musical segments for consideration as phrases. Given a chunk of music *with* tonal motion, however, the notion of “phrase” is still a bit fuzzy. Rothstein notes that some phrases may sound more complete than others, and two phrases may combine to form a larger phrase exhibiting stronger tonal motion.

While Musicat does not try to simulate the theoretical notion of “phrase”, these ideas do inform its notion of what constitutes a strong *group*. Rothstein contrasts phrases with other similar-length segments (*e.g.*, 4, 8, or 16 measures, *etc.*) that are still important units but that do not have the tonal motion required in his definition. He calls some of these segments “hypermeasures” if they exhibit rhythmic regularity and an alternation of strong and weak measures. Hypermeasures are a rhythmic, not a tonal, phenomenon. A set of successive four-measure-long hypermeasures contributes to a feeling of a steady pulse analogous to the “1–2–3–4” beats within a measure, but at a higher (slower) level.

The Structure of a Phrase (Motive, Period, Sentence)

Rothstein also discusses the smaller unit of the “motive” (I use the equivalent term “motif” elsewhere but will retain Rothstein’s word choice in this section.) He contrasts two different concepts of motive, given by Arnold Schoenberg (1967) and Hugo Riemann. Schoenberg states that “The features of a motive are intervals and rhythms, combined to produce a memorable shape or contour which usually implies an inherent harmony.” A motive for Schoenberg can be quite short, and connecting two or more motives can yield a phrase (although Schoenberg’s definition of “phrase” does not involve ending with a strong cadence, as required by Rothstein’s definition).

Schoenberg then describes two ways in which his phrases can be combined to yield larger structures: the period and the sentence. The difference between the two is not a question of length but of structure. A period is made up of a pair of phrases that are linked: typically, each phrase has a similar beginning, and there is a “question” or tension created by the first phrase that is resolved by the second phrase. The first phrase of such a pair is called the “antecedent” phrase and the related phrase that follows is called a “consequent” phrase. For instance, an eight-measure period might have one of the following forms, where each letter corresponds to two measures:

A B A C or A B A' C

A sentence, on the other hand, has an initial repetition (or near-repetition) of the first segment, which is followed by a segment of twice the original length that develops the initial motive, such as:

A A' B

where **A** and **A'** are two measures long but **B** is a four-measure unit. The segments are in the ratio 1:1:2.

Riemann also describes a process of combining motives into phrases and periods. (Rothstein, 1989, p. 27). However, according to Rothstein, Riemann's motive is primarily rhythmic in nature, where at each level of structure (motive, phrase, and period) we find an unaccented segment followed by an accented segment. That is, an antecedent–consequent relationship is found at every level. This seems extremely strict, but perhaps it captures many common musical structures, and it is interesting that rhythm (especially hypermeasure rhythm) is of such central importance here, as opposed to pitch.

Accent and Meter

I have been asked several times whether I have considered adding accents to Musicat's input; that is, letting it “listen” to something more like a human performance of a score, rather than a plain, uninterpreted score in which all notes have the same loudness. While this would be possible, I would find it less interesting than the current setup that uses unaccented notes, because a human listener should be able to understand metrical structure quite well (at least for simple enough music) even with a very plain performance. Rothstein seems to be in agreement:

This is perhaps an appropriate point at which to dispel a common misunderstanding concerning the nature of musical meter in general and of hypermeter in particular. It is sometimes thought that metrical patterns are established by means of dynamic accents — by singing or playing certain notes more loudly than others. This is not a necessary condition for the establishment of meter, as an increasing number of theorists have come to realize. In particular, the common view reflects a seriously impoverished conception of musical accent.

Later, Rothstein summarizes Carl Schachter's ideas of meter and accent, and writes: "The beginning of each new span [of time] *receives* an accent in the mind of the listener, just by virtue of its position in time; therefore it need not be *given* an accent by the performer..." Schachter himself writes:

...these accents result from the heightened attention attracted by the boundary points of the spans... [E]qual divisions, once established, can persist in the listener's consciousness without special sensory reinforcement. Indeed, they can persist for a time in the face of strongly contradictory signals...

Successive Downbeats

Typically, we expect measures of music to alternate in metric-hierarchy strength between weak and strong, and this downbeat–upbeat alternation occurs at each hierarchical level (as suggested by Riemann; see above). However, this alternating pattern is sometimes disrupted, and two successive measures may be heard as “downbeats” in terms of the hypermetric structure. Rothstein gives four conditions where this might happen (p. 58):

1. Metrical reinterpretation occurs, in a two-bar hypermeter context;
2. A hypermeter is contracted by dropping a bar (*e.g.*, a four-measure hypermeasure may be shortened to three measures);
3. If a phrase ends on a hypermetrical downbeat, the next phrase might start with another hypermetrical downbeat;
4. If the music consists of melody and accompaniment, then a downbeat might occur in the accompaniment, followed directly by a downbeat in the melody.

How to model the interpretation of hypermetric structures became an important issue in my work on the Musicat program. Musicat, as well as other models of music cognition, could likely benefit from careful consideration of Rothstein's work (the previous four conditions, for example, have not been incorporated into Musicat, but perhaps they should be in the future.)

The previous work described in this chapter was relevant to the aspects of the Musicat program concerned specifically with the domain of music. The remainder of this chapter discusses work not on music specifically, but rather on cognitive modeling in a variety of domains.

FARG

My program draws upon some three decades of work by Douglas Hofstadter and his graduate students and other colleagues in the Fluid Analogies Research Group (FARG). Musicat borrows ideas and architecture components from many previous FARG projects. In the following sections I describe some of the most relevant and influential features of each FARG project instead of giving in-depth descriptions. For detailed summaries of many of them I refer the reader to specific Ph.D. theses or to the book *Fluid Concepts and Creative Analogies* (Hofstadter and FARG, 1995).

There are two major features common to all FARG projects: the use of a very restricted domain and an architecture that supports fluid, creative perception. Projects developed by FARG have made use of *microdomains* to study cognitive mechanisms (Hofstadter 1995). Microdomains are very carefully stripped-down domains involving a much smaller problem space than the typical "real-world" domains seen in fields such as

artificial intelligence and machine learning. Problems in these domains, however, are sufficiently rich and deep so that arriving in a human-like fashion to solutions to them requires extremely flexible cognitive behavior. I call the architecture used to model such perception “the FARG architecture” for simplicity, although each FARG project has made its own unique variations on the central theme of simulating parallel subcognitive actions that respond to the conflicting internal pressures of human cognition. In FARG architectures, mental representations are formed, rejected, modified, and eventually settled upon in a flexible and stochastic way. Creative perception in these models is an emergent result of a great deal of seething subcognitive behavior.

I have found the study of many FARG programs helpful in my work on Musicat. Of particular interest are: Copycat, Metacat, Numbo, Tabletop, Letter Spirit, Phaeaco, Seek-Whence, Seqsee, Seek Well, and Cappyblanca. The authors of each program are listed below in the headings, but I have omitted Douglas Hofstadter’s name because it should be implicit: he was involved in each one as most of these programs were Ph.D. projects of his graduate students.

COPYCAT (MELANIE MITCHELL)

Of all the FARG programs, Copycat (Mitchell, 1993) has been probably the most influential for Musicat, because its architecture forms the basis for most of the FARG programs that came later. Copycat involves letter-string analogy problems such as:

“If **abc** goes to **abd**, what does **ijk** go to?”

Most people will answer **ijl**, seeing that the rightmost letter in **abc** was “incremented” and replaced with the next letter in the alphabet, **d**. Applying this rule to **ijk** is trivial. Now consider the following similar-looking problem:

“If **abc** goes to **abd**, what does **xyz** go to?”

The Copycat microdomain is deliberately restricted to a linear (non-wrapped) alphabet, and so the solution involving “wrapping around” from **z** to **a** is disallowed. Subtler analogy-making and greater conceptual fluidity are required to answer this question in a satisfactory manner. Many answers (providing varying degrees of aesthetic satisfaction) are possible besides the forbidden **xya**, such as **xyd**, **xyy**, and the surprising **wyz**. Analogy-making challenges generally do not admit of a single correct response, and it is the process of coming up with an answer, rather than the answer itself, that is the primary object of study.

Copycat’s architecture involves three main components: the Slipnet, the Workspace, and the Coderack, as well as a global temperature. I included a Slipnet in the first version of Musicat, but I removed it in the present version (see Chapter 9).

Slipnet

The Slipnet bears some resemblance to a semantic network such as Wordnet (G. A. Miller, 1995), in that it is a network of connected nodes, where nodes represent (the cores of) concepts. Copycat’s Slipnet includes approximately 60 nodes: one node for each letter of the alphabet, as well as nodes for concepts such as *letter*, *successor*, *predecessor*, *rightmost*, *leftmost*, *sameness*, and *opposite*. Nodes are connected by links, but unlike in traditional semantic networks, each link has a “length” representing the conceptual distance between the nodes. Each node also has an activation level, which varies dynamically as the program runs. The more active a node is, the more it affects the operation of other components of the

system. For instance, an active node can cause the program to search in the Workspace for more instances of the concept associated with the node. Active nodes also cause activation to spread through links (as in a connectionist network) to nearby nodes.

Workspace

The Workspace is the locus where perceptual structures are created, modified, and destroyed; I think of it as a model of human working memory. In Copycat, the three strings of the given problem (such as **abc**, **abd**, and **xyz**) are permanently present in the Workspace. Additional structures are dynamically formed based on concepts found in the Slipnet, such as a bond between the letters **c** and **d**, or groups of adjacent letters, a suggested answer string, and so on. Higher-level structures such as meta-groups made up of groups can emerge as more structures are built in the Workspace.

The activity that takes place in the Workspace is highly fluid. Small bits of computer code called “codelets” (described in the next section) are the source of this activity: many codelets run in a highly parallel fashion, modeling subcognitive activity that is quite unpredictable at the micro scale, as different concepts in memory are activated and perceptual attention rapidly shifts between objects. This activity may look like a seething, chaotic mess: codelets can build and destroy structures in the Workspace, and probabilistic “competitions” between codelets cause structures to flicker in and out of existence rapidly. However, over time, stable structures *emerge* as a result of these interactions. Through the activity of generating structures and destroying rival structures, Copycat performs a sort of heuristic search through possible pathways toward solutions, which is termed the *parallel terraced scan*. Instead of being a brute-force search of the sort familiar in artificial intelligence, or even a heuristic-based search such as A^* , Copycat’s search process is a much more

cognitively plausible simulation of how human thought might work. For example, Copycat can exhibit phenomena such as getting “stuck” in a problem, or having a “Eureka!” moment when a new, much more satisfying cognitive structure is formed which solves the problem. These are both examples of the flexibility of Copycat’s architecture.

Coderack, Codelets, and Pressures

As mentioned above, structures in the Workspace are created by *codelets*, which are small chunks of computer code representing subcognitive activity. Codelets do the work of creating, destroying, modifying, and perceiving structures in the Workspace, and they also affect activations of nodes in the Slipnet. The *Coderack* is a staging area where codelets, after being created, wait until they are selected to run. Some codelets are created and posted to the Coderack by active concepts in the Slipnet, and others are spawned by other codelets.

But why does Copycat use codelets to model perception? Central to Copycat’s architecture is the notion of subcognitive *pressures* (*i.e.*, biases, often context-dependent, that affect a host of small decisions made as the program runs), and the codelets on the Coderack are one of the ways in which these pressures are implemented (Slipnet node activations are another of several sources of pressures spread throughout the architecture). Although there are a great many stochastic decisions made during the course of a run and the program’s behavior may seem quite chaotic when viewed over a short time scale, various pressures in the program serve to guide the course of a run so that over a longer time scale, the program’s perceptual activity has more direction (even though it is still nondeterministic).

Each codelet has an urgency value, so that a high-importance action (such as that recommended by a highly active Slipnet node) can be marked as highly urgent, making it likely to be selected more quickly. The Coderack uses these urgency values as well as information about the current global “happiness” of objects in the Workspace as biases in the

stochastic selection of which codelet to run next. (The number of codelets of each type on the Coderack constitutes another source of pressure.) This allows the program to explore multiple different cognitive pathways (the parallel terraced scan mentioned above) at different speeds.

Temperature

The Workspace is in a constant state of flux, with groups and links at many levels of structure being created and destroyed all the time in the search for strong stable perceptual structures. Codelets compute a measure of “happiness” for each perceptual structure, indicating how stable it is and how consistent it is with associated structures. As the overall amount of happiness increases, the system’s global temperature drops. The temperature affects how urgencies are used in selecting which codelet to run next, it affects the probabilistic choices that are made by individual codelets, and it also affects the types of codelets added to the Coderack: when temperature is high, more “breaker” codelets are posted to break up existing structures to make room for new ones, whereas when the temperature drops, codelets tend to behave collectively in a way to make the Workspace more stable.

Temperature varies inversely with the overall Workspace happiness, but it also gradually drops over time in order to force the Workspace to stabilize. This controlled descent of temperature is reminiscent of simulated annealing, although it is not deterministic or monotone: a very low level of happiness in the Workspace can cause the Temperature to jump back up to a high level.

METACAT (JAMES MARSHALL)

Metacat, the successor to Copycat, added a degree of self-watching to Copycat's architecture. For example, whereas Copycat might get "stuck" re-discovering the same unsatisfying structures multiple times in a single run, Metacat has some ability to notice the repetition and purposefully avoid it. I haven't used any of these particular new mechanisms from Copycat, but the general idea of self-watching is important and comes into play in one aspect of my program, which was also inspired by Tabletop (see below).

Metacat introduced many new ideas not present in Copycat. In addition to the components present in Copycat such as the Workspace, Coderack, and Slipnet, Metacat's architecture also includes an "Episodic Memory" — which augments Copycat's long term memory, and it also includes a "Temporal Trace", and "Thespace", which provide a meta level that allows self-watching. For more detail please see Marshall's thesis — in this section and in the description of other FARG programs, below, my intent is not to describe the programs in depth, but rather to give credit to particular aspects of these programs that inspired me for one reason or other. In general, I refer the reader to the relevant books or dissertations for more information on each of these programs.

NUMBO (DANIEL DEFAYS)

Numbo is one of the simplest (but most elegant) FARG projects. It makes obvious the difference between raw, bottom-up perception and more directed, top-down perception. As is the case with Metacat, I did not use any of Numbo's elements explicitly, but I found the ideas in it interesting and helpful in clarifying my thinking.

The Numbo program solves puzzles in which a set of positive integers, called "bricks", is provided along with a target positive integer. The goal is to find a mathematical

expression that equals the target. Such expressions are built up using the bricks and a small set of arithmetic operators (+, −, ×) and possibly parentheses. For example:

Bricks: **5, 3, 13, 2, 8**

Target: **42**

One solution is the expression $8 \times 5 + 2$, although others are possible⁵. Of course, solving the puzzle would be easy for a brute-force search algorithm, but the point of Numbo is to solve the puzzle this in the same way as humans do. Numbo has preferences to notice certain arithmetic properties of numbers and ways of combining them that are salient for people. In this example, the arithmetic fact $8 \times 5 = 40$, in the context of the bricks and target above, might be much more salient than the fact $50 - 8 = 42$. Thus it might be less likely for the program to come up with the alternate solution $(13 - 3) \times 5 - 8$.

Numbo, like Copycat, uses a memory of permanent knowledge with spreading activation called the “Pnet” (essentially a simpler Slipnet), a Workspace, and a Coderack with codelets. During a run, bricks are combined stochastically in the Workspace to form new blocks. For instance, **8** and **5** might be combined quite easily using multiplication to form a block with value **40**, and the proximity of **40** to the target **42** would strengthen the block. On the other hand, blocks such as $13 \times 8 (= 104)$ with no obvious relationship to the target would be less likely to form. Although it would be possible for such a block to form as a result of the constant stochastic activity of the system, if formed it would likely be quickly broken apart by “dismantler” codelets.

⁵ Other solutions are $3 \times 13 + (5 - 2)$ and $(3 \times 13) + (8 - 5)$. The solution above, however, was the most obvious to me, because it is less natural to me to multiply by **13**. However, someone who saw the target, **42**, as 3×14 might realize that it is possible to get close to the target with 3×13 , leading to these solutions.

The most interesting aspect of Numbo for me is the way that while it has a significant stochastic aspect, allowing blocks to form and disintegrate quite wildly, it also acts in a goal-oriented way, despite the randomness. This is true of all the programs based on the FARG architecture, but I found that this aspect of the architecture even more evident to an observer in Numbo than in other FARG programs. Also, in this domain it seems particularly easy to see how FARG programs differ from programs using classic heuristic search, such as is used in the General Problem Solver (Ernst & Newell, 1969):

One would be at pains to cast Numbo's method of proceeding in terms of search in a problem space; after all, no central control is involved, and rote knowledge is represented in a task-free way. A goal is just one of many features of a situation. No systematic exploration takes place in Numbo. Taken together, competition between codelets and the element of randomness allow the system to jump from one idea to another, sometimes in a chaotic-seeming (but rather humanlike) way. (Defays, in Hofstadter & FARG, 1995, p. 149)

In Numbo, as other FARG projects, “search” is replaced with stochastic juggling of mental objects, pushed by pressures to examine “interesting” combinations. This mental manipulation often results in good ideas and solutions, but sometimes fails to find a satisfying answer, but sometimes finds a surprising or even a quite aesthetically-pleasing answer — much as happens in human cognition.

TABLETOP (ROBERT FRENCH)

The Tabletop program solves problems in a surprising and initially silly-sounding microdomain: touching objects on a coffeehouse table. A problem starts with a representation of the two-dimensional surface of a table, with various arrangements of silverware and cups and glasses and such items on either of two sides of the table in particular locations. We imagine that two people (Henry and Eliza) are seated across from each other at the table. The

task is simply this: Henry touches an object on the table, and Eliza (the program) is supposed to mirror this action by “doing the same thing”. For example, if each side of the table has a cup and a spoon, and if Henry touches the cup on his side of the table, Eliza would probably touch the cup on her side. This seems very simple, but the challenge quickly grows very tricky and subtle when different objects in different arrangements are present on the two sides. The deep problem attacked by the Tabletop project is actually context-dependent analogy-making between groups of objects.

Tabletop’s architecture is closely related to that of Copycat, but it introduces many new ideas to handle difficulties that cropped up in the Tabletop domain. The new ideas include: less-destructive competition between structures, a solution to the “urgency explosion” in the Coderack, and a more sophisticated computation of temperature.

Urgency Explosion

While he was developing Tabletop, French noticed that “one codelet of a given urgency can spawn many codelets of the same (average) urgency” (French, 1992), which resulted in a sudden decrease in relative probability of being selected for other codelets on the Coderack. This had not been a problem in Copycat because there, unlike in Tabletop, each codelet could spawn only one follow-up codelet. French’s solution was to compute an adjusted urgency that decreased exponentially with the number of generations that had passed since an original ancestor started spawning descendant codelets. In other words, fresh, new codelets representing newly-generated pressures or newly-perceived elements of the scene were favored over stale, old, and potentially out-of-date codelets.

Less-Destructive Competition

In Copycat, structures that lost competitions with other structures were destroyed instantly. This strategy was satisfactory for Copycat and Numbo, where the structures in the Workspace were rather small and could be easily reconstructed stochastically as long as the proper pressures were present. In contrast, Tabletop allowed losing structures to remain in the Workspace, although they “faded out” a little bit. The Tabletop Workspace was thus a collection of many overlapping structures of various strengths, even though many of the overlapping structures in this Workspace would have been mutually exclusive in a Copycat-style Workspace. A new architectural component called the *Worldview* was introduced by French, which maintained a single non-contradictory mapping between objects and groups on the table. Old “faded” structures in the Workspace were allowed to compete to return to the Worldview. This obviated the need to reconstruct large structures from scratch, which would have been extremely unlikely, and provided a more psychologically plausible way for large structures to compete with each other (in a way that is reminiscent of the Necker Cube or Vase/Faces illusions.)

Temperature

Tabletop’s calculation of temperature differs from Copycat’s in that Tabletop’s temperature acts more locally and “indirectly”. It takes into account the strengths of structures in the Worldview, but ignores some objects completely, in much the same way that Capyblanca (see below) ignores portions of a chessboard. The factors involved in Tabletop’s temperature computation are:

1. quality of the Worldview, composed of the following:
 - sum of the strengths of all correspondences;

- degree of coherence in how correspondences “bunch together”;
 - degree of parallelism between correspondences;
2. average rate of change in Worldview quality;
 3. number of changes in the Worldview;
 4. number of competitions.

The first factor has the largest effect on temperature. The other three regulate the temperature in a way that provides simulated annealing, and, indirectly a limited kind of self-watching.

Of particular interest is how the notion of “slippage” was handled in Tabletop. Whereas in Copycat only one overall analogy was being formed, Tabletop might have two or more unrelated sets of correspondences, and concept slippages in one set might be unrelated to the other. However, there is only one copy of each Slipnet node, even though in the case of multiple, simultaneously present correspondences, it seems that multiple copies of each node might be necessary. French refers to this problem of the Slipnet’s state being applied to wildly different contexts as “the problem of single nodes with multiple activations”. A particular Slipnet state might be relevant to one part of the problem (such as one spatially isolated location or one particular level of grouping hierarchy), but not to another part of the problem. He considers possible solutions to this dilemma, and settles on the idea of reconstructing activations of Slipnet nodes each time the program switches from one context to another.

LETTER SPIRIT (GARY MCGRAW AND JOHN REHLING)

The program Letter Spirit aimed to model “the creative act of letter-design” (Hofstadter & Fluid Analogies Research Group, 1995). Although I didn’t use any specific

features of Letter Spirit in my work, I was influenced (as mentioned in Chapter 1) by the program's focus on modeling creativity. Letter Spirit has separate modules for designing new letters and for examining previously-designed letters, and this separation of creation and perception was appealing to me. Musicat does not create music, but it does perceive music, just as one module of Letter Spirit was designed to perceive letter shapes.

PHAEACO (HARRY FOUNDALIS)

The Phaeaco program solves Bongard problems, which are visual puzzles that require the puzzle-solver to find a visual concept that is exhibited by each image in a set of six images, and that is not exhibited by another set of six images that are known to be non-examples of the concept. The most significant idea in Phaeaco that I incorporated in one version of Musicat was the idea of concept formation in long term memory, including the ability to persist the long-term memory across multiple runs of the program. Phaeaco can learn visual concepts and recall them while solving future problems. One version of Muiscat (not the latest version, however) incorporated this notion and could learn musical motifs and then recognize them in future runs.

SEEK-WHENCE (MARSHA MEREDITH) AND SEQSEE (ABHIJIT MAHABAL)

The program Seek-Whence, and its descendant, Seqsee, were designed to understand sequences of numbers and to make predictions about which numbers would come next in the sequence. For instance, consider the following sequence, from *Fluid Concepts and Creative Analogies* (Hofstadter & Fluid Analogies Research Group, 1995):

1 2 2 3 3 4 4 5 5 6 ...

This sequence might be understood as consisting of the number “1” followed by a series of pairs of numbers, continually increasing, as so:

$$1 \ (2 \ 2) \ (3 \ 3) \ (4 \ 4) \ (5 \ 5) \ (6 \ \dots)$$

However, the initial “1” stands out as a “glitch” in the pattern, so people (or the programs Seek-Whence and Seqsee) might prefer to see the pattern as a series of groups in which each group consists of a number followed by its successor, as follows:

$$(1 \ 2) \ (2 \ 3) \ (3 \ 4) \ (4 \ 5) \ (5 \ 6) \ \dots$$

When I was first reading *Fluid Concepts and Creative Analogies*, I read some of the number sequences at the start of the book and imagined hearing them as sequences of musical notes (imagine mapping the number “1” to the note “C”, “2” to “D”, and so on. Therefore, I was please to read Hofstadter’s comment that this second parsing of the number sequence above “sounds like” a Chopin piano prelude: Op. 28, No. 12 (Hofstadter & Fluid Analogies Research Group, 1995, p. 79).

Seek-Whence and Seqsee gave me great confidence that I could write a program that used similar techniques as they did, but in the domain of music, because of the number-to-note mapping mentioned above. The program (Musicat) would require knowledge specific to the music domain (the notes “C” and “G” are related in a special relationship that does not exist between the numbers 1 and 5, for instance). Also, Seek-Whence and Seqsee incorporate the notion of expectation, which was especially important in the early versions of Musicat.

Several specific features of Seqsee were influential in Musicat’s development. Seqsee made less use of the idea of a Slipnet than most earlier FARG programs, and in the latest version of Musicat I did away completely with the Slipnet. Seqsee introduced a novel “stream

of thought”, and although I did not use this idea in Musicat, I found the idea appealing and it subtly influenced how I thought about the constant progression of musical time. Finally, the user interface of Seqsee, with bar graphs showing the distribution of various codelets on the Coderack, inspired some of the graphs I made during development.

SEEK WELL (STEVE LARSON)

Steve Larson implemented two versions of his Seek Well program — the single-level model and the multi-level model described earlier in this chapter. However, he also gave a set of instructions for implementing his theory of musical forces more completely using a FARG-style architecture. His instructions describe in some detail how the model should generate predictions of melodic completions by automatically generating a hierarchical embellishment analysis including information about the key, mode, and meter, and using this analysis to determine reference alphabets and goal alphabets and to make predictions based on the three musical forces. Larson writes that a complete implementation of his model, based on a FARG-style program that takes all the instructions into account in parallel, “will most likely be quite informative about music cognition” (Larson, 2012).

This parallelism is critical; without it the model is limited. Larson’s own implementation of his multi-level model works only in a serial fashion, generating predictions at the deepest structural levels first. However, there are cases where the surface levels can have a more direct impact on predictions. In general, musical forces working at different structural levels can conflict with one another, which shows the need for the model to be able to deal with such conflicts when making melodic predictions. Parallelism would provide a way to handle these conflicts. Additionally, Larson’s experimental work shows how other important musical factors such as implied metric structure and implied harmony affect how a human listener hears a melody and generates possible extensions of it; these, too, could

be considered in parallel with other factors. Finally, Larson's implementation required as input not only a melody, but also a human-generated embellishment analysis thereof. The model did not provide a way to generate the necessary embellishment analysis automatically, but ideally it would also be generated in parallel with the rest of melodic processing and expectation-generation. Larson writes:

A complete specification of the theory might describe how hierarchical representations of embellishment structure may be built up and internally represented. (Larson 2004)

My initial version of Musicat (also called "Seek Well") was an attempt to implement a FARG-style model according to Larson's theory of musical forces. However, I ran into obstacles caused by the need to automatically generate the embellishment structure (this is essentially the problem of generating an automatic Schenkerian analysis). Although I believe that a FARG-style architecture would prove useful in generating such an analysis, I removed the embellishment-structure code from my program early during the development of Musicat (see Chapter 9); this important part of Larson's theory of musical forces is out of the scope of my project, which focuses more on grouping structure. Similarly, the initial determination of key and meter, which was a key focus in Larson's Seek Well domain, is not considered in the current version of Musicat, which is given that information "for free". Some of the ideas from Larson's work on Seek Well do appear in Musicat, however, such as the reference and goal alphabets described above, and the general idea of allowing multiple musical interpretations and expectations to compete with each other.

CAPYBLANCA (ALEXANDRE LINHARES)

Computer mastery of the game of chess was once a sort of Holy Grail of artificial intelligence research, but Deep Blue's triumph in 1997 over world champion Garry Kasparov using brute-force lookahead search may be seen as a hollow victory for AI: Deep Blue's success in playing chess has nothing to do with playing chess in the way people do. A computer employing this sort of search strategy may play an extraordinary move (such as a wild-looking queen sacrifice), but it would not find the move any more exciting or surprising than any trivial defensive pawn move. Human masters sometimes play spectacular moves for fun when a more modest move would win a game more quickly. Ironically, while people are able to appreciate the aesthetic value of computers' brilliant chess moves, but the computers themselves remain totally oblivious. Today, computer chess feels pretty much like a "solved" problem to the AI community, and interest has turned to the game of Go, in which humans are still far superior to computers (just as was the case for chess through the mid-1980s), although some groups are making rapid progress.

There is still reason to consider chess as an interesting domain of cognitive-science research, however. Alexandre Linhares used a FARG-style architecture to model perception in the game of chess (Linhares, 2008). His program, called Capyblanca (a reference to former world champion José Raúl Capablanca), is unusual for chess computing in that it does not carry out a brute-force lookahead search of the game tree. Instead, it uses ideas about chess intuition (Linhares, 2005) to simulate how a human player might visually scan the chess board, noticing salient aspects of the position, remembering relevant positions from previous games, and forming a long-range plan. Capyblanca not only constitutes an alternative to chess programs such as Deep Blue, but it also challenges some traditional thoughts on expert knowledge in chess (Linhares & Freitas, 2010). Note that the current version of Capyblanca does not actually make chess moves; instead, it simply perceives salient aspects of chess

positions and forms large-scale plans. Perhaps a future version of Cappyblanca will play human-like chess games.

Cappyblanca offers a way to model the following features of human chess:

- focusing on salient features in a position;
- investigation of key moves, avoiding brute-force search;
- recognition of strategic themes from past games;
- generation of broad expectations for upcoming game positions;
- conscious ideas emerging from subcognitive processes.

Cappyblanca's architecture was inspired by earlier FARG projects such as Copycat and Phaeaco, but it has some unique aspects, including the following:

- **A new way to calculate temperature for a structure.** This involves relevance to current goals, ability to imagine a structure (such as a queen in a position where it isn't on the current board), and the "evidence" for a particular structure's existence. Temperature is locally computed for each structure, as in Tabletop.
- **Starting from an empty Workspace.** The Workspace in Cappyblanca starts out completely empty, and perceptual structures describing the relevant features of the chess position are gradually built up at several different levels of description in parallel, based on what the program "looks" at (see next bullet). In most other FARG projects, the Workspace starts out including input elements (in Seek Whence, Seqsee, and Musicat, the Workspace is

initially empty but input elements are automatically added to the Workspace over time).

- **Simulation of eye saccades.** Whereas most chess programs maintain a list of all the pieces on the board, Capyblanca “sees” only certain pieces, depending on random simulated eye movements. Its focus of attention can move randomly and can also be attracted to nearby vaguely-perceived structures or salient parts of the board. This process of shifting attention is quite human-like and constitutes a “visual” top-down pressure to focus on certain aspects of a board rather than on others.

CHAPTER THREE

Musicat's Domain

Introduction

Each program based on the FARG architecture — Copycat, Tabletop, Letter Spirit, Phaeaco, Seqsee, and so on — operates in its own microdomain, as was discussed in the previous chapter. The use of *microdomains* in these models is not only humble and realistic, it also helps programs focus on particular aspects of cognition without being overwhelmed by real-world domain complexity. This chapter describes Musicat's microdomain and gives examples of the types of music the program is intended to handle. In brief, the microdomain is made up of simple, singable Western melodies, including folk tunes such as “Twinkle, Twinkle, Little Star” and “Row, Row, Row Your Boat”; popular melodies such as “On the Street Where You Live” (from the musical *My Fair Lady*) are also part of the domain, albeit on the more sophisticated end of the spectrum of what Musicat is meant to “listen” to.

Although the spirit of FARG projects is to work in microdomains, for the sake of musical interest we have expanded Musicat's microdomain over time enough that the word “micro” seems less applicable than it did for previous FARG programs. Therefore, sometimes (as in this chapter title) I omit the “micro-” prefix and simply write “domain”. Compared with music in general, however, this domain is certainly “micro” — just think of the

complexity of Bach's Inventions, not to mention his *Well-Tempered Clavier*, or Monteverdi's Madrigals, Chopin's Nocturnes, Shostakovich's Preludes and Fugues, Mahler's Symphonies, Rachmaninoff's Piano Concertos, Wagner's Operas, and so on and so forth. And if we expand this list further to include atonal music, contemporary popular music, jazz, musical theatre, minimalism, electronica, or even different musical traditions entirely such as Balinese *gamelan*, Bhutanese *zhungdra*, or Brazilian *choro*, we see that music is almost impossibly diverse even for a human listener — let alone a computer model — to understand.

A NOTE ON THE UNIVERSALITY OF MUSICAT'S DOMAIN

Musicat is designed to simulate the experience of listening to relatively simple, monophonic, Western, tonal melodies; it does not attempt to handle the wide variety of musical styles listed above. In this sense, Musicat operates in a very limited musical space. Much of the complexity of listening to music, however, is encountered even in simple-sounding melodies, so Musicat's domain is still highly complex, the specific restrictions described in the next section notwithstanding.

What is it about modeling the listening process in Musicat's domain that may be applicable to modeling listening for other musical styles? Certainly, some aspects of Musicat's approach are style-specific. But in general, the approach should transfer well, because it is based on many features that all types of music-listening have in common, including music's temporal nature, human memory systems, gestalt principles, and the centrality of analogy-making. Anything we call music is based on structured sound events arranged in time, so Musicat's handling of issues of temporal perception should be applicable to all types of music. Human memory systems are, obviously, implicated in all music listening (and in general all of cognition). Gestalt principles that inform perception of musical groups and

musical expectations also typically operate at the lower levels of perception, and should cross over to most musical contexts. Finally, analogy is central to Musicat’s listening strategy, and if analogy is the core of cognition, as Douglas Hofstadter argues, then it follows (trivially) that analogy is also the core of *music* cognition (see Chapter 4). Musicat is better equipped to listen to extremely simple Western melodies such as “Twinkle, Twinkle, Little Star” for the time being, and it incorporates domain knowledge that is specific to Western tonal music, but its foundations should apply to music-listening in general.

Microdomain Specifics

MELODY CHARACTERISTICS

Musicat was originally conceived as an extension of Steve Larson’s Seek Well project. It has diverged significantly since the first version (as described in Chapter 9), but it will be helpful here to explain how Seek Well influenced Musicat’s microdomain. Indeed, Larson used “Seek Well” not only as the name of his program, but also as the name of the microdomain itself, as in “the creative microdomain Seek Well” (Larson, 1997a). Melodies in the Seek Well microdomain followed the set of rules described in Chapter 2. Musicat’s domain is less restrictive, in the following ways:

- Only one note sounds at once, just as in Seek Well. In early versions of Musicat a simple bass line was allowed to describe chord progressions, but this is not done in the most recent version.
- Notes can have the following durations: whole, half, quarter, eighth, sixteenth, and dotted versions of each of these except for sixteenth. Notes can be tied to immediately following notes. Note that notes shorter than

sixteenths, triplets, and other more complex types of durations are not supported.

- The only attributes that distinguish one note from another are *pitch* and *duration*. In earlier versions of Musicat, notes could be explicitly accented in the input.
- Rests are allowed, of any of the durations available for normal notes.

EXTRA INFORMATION INCLUDED WITH MELODIES

One of the interesting features of the Seek Well microdomain was that key signature and time signature were not specified; programs had to infer these properties themselves (just as human listeners do). For simplicity, we provide key and time signatures to Musicat. The metric position of the first note, which may be an upbeat, is also provided (see Chapter 8 for more details). The key and time signature must stay the same throughout the melody.

In a similar vein, note that Musicat is given musical input in a symbolic form much like sheet music. The program knows from the outset that it is hearing a “quarter-note C” or “whole-note F”; it doesn’t listen directly to sound waves, and no audio processing via Fourier transforms or their ilk is required. Audio analysis at these lowest levels is out of the scope of this work.

TWO EXAMPLES

The next section gives detailed descriptions of what it is like for humans to listen to melodies of the sort expected by Musicat. As a preview, here are two of the melodies, which abide by the rules listed above.

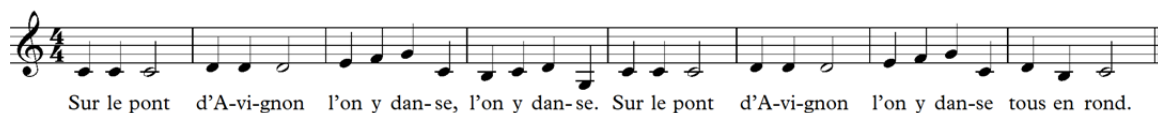


Figure 3.1: “Sur le pont d’Avignon”, a typical melody in Musicat’s domain.

The French folk tune in Figure 3.1 is a typical melody in Musicat’s domain. The melody is relatively simple both rhythmically and melodically. The rhythm is made up of just quarter notes and half notes, and the melody is tonal and easy to sing. Note that the lyrics are certainly not part of the domain, and they are not given to Musicat. I include lyrics in the figures simply to aid the reader in remembering the melodies.



Figure 3.2: “On the Street Where You Live”, a more complicated melody in the domain.

The melody from “On the Street Where You Live” (Figure 3.2), as was mentioned previously, is a more sophisticated melody in this microdomain. Only the first 16 measures of the melody are pictured here; the full melody is much longer. Note the use of ties across bar lines. There are just two beats in the first measure (a “pickup measure” containing two “pickup notes”, or “upbeats”), but the rest of the measures, except for the final measure, have four beats each. Not pictured explicitly is the more complex tonal structure of the melody. Musicat “hears” melodies as if they were performed *a cappella* — no chords are given — but melodies tend to imply harmonic structures. The implied harmonies in this example are more complicated than those in Figure 1. The grouping structure of the melody is also more sophisticated. (An exercise for the reader: where do you hear groups starting and ending in this melody?)

Example Melodies and Human Listening Performances

In this section I discuss three melodies (including the two melodies above) from the point of view of human listening. Chapters 6, 7, and 9 will examine Musicat's performance on a variety of melodies including these three, but for now I focus on how I, as a human listener, hear these melodies. Some of the things I describe in this section can be modeled by Musicat, but some can't.

To be clear, these are introspective walkthroughs of my own subjective experience of listening to these melodies, and are not based on EEG, fMRI, or other laboratory methods that some readers might expect. If this sounds suspect for some reason, please realize that I am simply describing my own experience in order to introduce the challenge of music-listening, in preparation for a careful examination of Musicat's behavior to follow in later chapters. Moreover, these melodies are simple enough that I am sure there is not much variation in how listeners who are experienced in Western tonal music will hear them, at least in terms of low-level grouping structure.

MYSTERY MELODY

For my first example, I will reproduce a melody that Steve Larson played for me in 2003 to demonstrate some aspects of listening. He played it one note at a time, and we stopped to discuss what I had heard and what I expected after each note. I will try to replicate the experience here, by adding a single note to each successive figure in the next several pages. I encourage the reader to try not to look ahead — we can't "look ahead" when listening, after all! — but instead, sing the melody to yourself and think about your own expectations. I will use the pronoun "we" in this section to bring you, the reader, along with me as we listen to this short melody. See the next figure for the first note.



Figure 3.3: Mystery Melody, Note 1.

We have heard a single note with a quite typical duration. A person with absolute pitch (also known as “perfect pitch”) might have heard this as a middle C — for the rest of us, it just sounds like a note with a pitch from somewhere in the middle of a piano keyboard (in the rest of this section I will use the note names for simplicity, but we can think of “D” as simply meaning “the note above the first one”; when I use the note names I am not meaning to suggest that typical listeners know the names of the notes they hear). Do we have any expectations about what notes might come next? It certainly doesn’t seem there is much information to go on. Let’s listen to the next note:



Figure 3.4: Mystery Melody, Notes 1–2.

The melody is now a bit more interesting — two notes are better than one! We have heard two notes that are close together in pitch: the second note (D) is a whole step higher than the first. What do we expect to come next? In my case, I expect the next note to be another step higher still; I expect to hear an E. (See Appendix B for a pilot study investigating expectations that are formed after two notes such as these are heard.) Another very reasonable expectation is for the melody to go back down to C.

By this point, I also have heard the first note not simply as “C” but also as “the tonic note of the scale” (I wouldn’t *consciously* think about the note C being the “tonic”, but I believe my internal representation of the first two notes would be somehow associated with

this concept). I have implicitly heard this as a melody in the key of C — more specifically, C major. Also, I have heard the first note as a downbeat; the C sounds stressed to me and the D sounds unstressed. Even when I have a computer play the two notes for me with equal volume, I hear the C as stressed (this entire melody should be performed with equal volume and articulation for every note; any stresses should not be present, but rather should be implied and ascribed to notes in the “mind’s ear” of the listener).

We have default assumptions that kick in rapidly as we listen, but the melody may turn out to prove us wrong. Nothing is forcing the melody to sound like C major; it might turn out to be in the key of A minor, for instance. There is also no guarantee that the first note is a downbeat; it might be a pickup note, with a downbeat D, and perhaps the melody would continue in D minor (imagining this hypothetical way of hearing the first two notes instantly reminded me of the song “Summer Love” by Neil Diamond, from the movie *The Jazz Singer*, whose melody starts this way: the first note in the orchestra is a lowered seventh scale degree, and the second note is the tonic, accompanied by a minor tonic triad).



Figure 3.5: Mystery Melody, Notes 1–3.

The third note makes me more confident in my default assumptions and expectations. Indeed, the E that I expected came next. This certainly sounds like the start of a C-major melody, and the first note sounds like it was indeed a downbeat.

Different people will have different expectations, but I expect the melody to continue up the scale at this point with F and then G. Maybe it will even continue up to the C an

octave higher. I could also imagine that the melody will turn around and come back down to C instead of going up, but my first expectation was for an F to come next.

What will come next? Will the melody go up to F or drop back down to D? Or will it drop immediately down to C as in the song (coming up in Chapter 6) “*Frère Jacques*”?



Figure 3.6: Mystery Melody, Notes 1–4.

Another E — what a surprise! I find this second E to be confusing. It is remarkable that this extremely simple-looking 4-note pattern causes surprise and confusion, but it does, at least for me. What do we expect to come next now? The second E seems to bring a halt to the upwards motion established by the first three notes (according to Larson’s theory of musical forces, we might say that the melody’s upwards-pointing inertia was overcome by the force of magnetism or gravity). I find it uncomfortable to stop for long here after the repeated note, so let’s listen to the next note.



Figure 3.7: Mystery Melody, Notes 1–5.

The melody has continued its upwards climb after a brief pause on E. But this is too simple a description of what the melody sounds like. At this point, many interesting things are happening (or have already happened) to me as a listener, although I will delay describing them in detail because I don’t want to ruin the experience for you, the reader. For me, the

melody is still a little confusing here, but after another note all will be resolved. What do we expect to come next?



Figure 3.8: Mystery Melody, Notes 1–6.

The previous two notes, E, and F, left me feeling confused for some reason, but by the time I heard the G, everything made sense. I hear these six notes as two copies of a 3-note pattern: C–D–E followed by E–F–G. The pattern is made up of three notes that are successive steps in the C-major scale. The first instance of the pattern started on C, while the second instance started on E (which was the same note that the first pattern ended on).

Perceptually, the elements of each pattern form a group: I hear the first three notes as being grouped together, and likewise the last three notes sound like a group. Furthermore, when I say that they sound like instances of a pattern, I am really making an analogy between the two groups: C–D–E is like E–F–G.

Why was I so confused by the repeated E? Besides the E denying my expectations for the motion to continue up or down, I was surprised that even though the two E's had the same pitch, and sort of sounded like they should be grouped together, they eventually turned out to be members of *different* groups. The first E turned out to be the end of a group, while the second E was the start of a different group.

The real surprise, though was that the groups I heard had three elements each. When I heard the initial notes C–D–E, I heard the C as a downbeat, the D as a weaker beat, and the E as a stronger beat. That is, I heard them in duple meter. Specifically, I expected the fourth note to be the last beat of a measure in 4/4 time, and I expected another strong

downbeat to occur at the point in the melody where we heard the F, as in the following figure, in which implied downbeats are marked with accent marks, and a smaller implied stress, typical in 4/4 meter, is indicated with a stress mark over the first E:



Figure 3.9: Mystery Melody, Notes 1–5, heard in 4/4.

However, the F doesn't sound like a strong downbeat at all! Instead, the second E sounds like a downbeat, like so:



Figure 3.10: Mystery Melody, Notes 1–5, heard in 3/4.

In short, I was surprised because I started off hearing the melody in 4/4, but over the course of two or three notes (from the first E through the G) I had to revise my hearing to hear everything in triple meter (3/4) instead (this shift in metric interpretation is similar to the one I described in Chapter 1 for the opening of the song “Ants Marching”).

The mental processes involved in listening to these six notes are representative of those that Musicat is intended to model. To hear these notes as two similar 3-note groups involves grouping, analogy-making, expectation-generation, and revision of previous perceptions.

After hearing the first six notes, what do we expect to come next? I recommend trying to sing the entire melody from the start and to improvise a conclusion. Will the melody fulfill our expectations?



Figure 3.11: Mystery Melody, Notes 1–7.

For me, this was just what I expected. Because I heard two 3-note groups, one starting on C and another starting on E, I expected the pattern to continue with a group starting on G and continuing upwards.



Figure 3.12: Mystery Melody, Notes 1–8.

The note A was just what I expected; nothing is surprising here.



Figure 3.13: Mystery Melody, Notes 1–9.

The B was also expected. At this point, I have heard three copies of the 3-note pattern, and furthermore, I have heard the music in triple meter, as in the following figure:



Figure 3.14: Mystery Melody, Notes 1–9, heard in 3/4.

Notice that at this point something subtle occurs in our expectations. It would be ridiculous to expect another copy of the pattern starting on B as follows:



Figure 3.15: An unrealistic expectation for the next three notes.

We don't expect the pattern to repeat, starting on B, despite the seemingly mathematical precision of the first nine notes. However, a computer program (such as Seqsee, for instance) might generate such an expectation if it "heard" the pattern in the following way, with pitches replaced by numbers with no musical meaning (expectation shown in parentheses):

1–2–3 3–4–5 5–6–7 (7–8–9)

Instead of this expectation, we hear the B as a leading tone that "wants" to resolve to the tonic (the C a half step above). Moreover, the downbeat notes C–E–G outline a C-major triad, and we expect motion through a C-major triad (*i.e.*, an arpeggiation of the C-major chord) to lead directly to the C on the next downbeat, even though it breaks the surface-level pattern established by the first nine notes.



Figure 3.16: Mystery Melody, Notes 1–10 (complete).

Indeed, this short melody continues (and concludes) as expected, on the note C. In my 3/4 hearing, the melody would be notated as follows (the accent marks, as before, simply indicate implied accents):



Figure 3.17: Opening theme of Bach's Organ Prelude in C major, BWV 547.

It turns out that our mystery melody is the start of a Bach prelude, from the Organ Prelude and Fugue in C major, BWV 547. Henceforth I will refer to this melody by a shorthand name, “Triplet Scale”, to emphasize its 3/4 nature. This melody is of the sort that early versions of Musicat were designed to listen to. However, Musicat evolved to focus on larger-scale musical structures and expectations, so the note-by-note description of my experience of listening to this melody is not indicative of what the latest version of Musicat does (but see Chapter 9 to learn more about the early versions of Musicat and to see it running on this melody).

Although this melody is very short and simple, there are complex cognitive phenomena involved in forming default expectations, in making groups and analogies, and in switching from a duple- to a triple-meter way of hearing somewhere between notes 4 and 6. These same sorts of complications arise in most music, so I find this melody to be quite inspiring and even provocative: what a challenge it poses for music cognition research, despite its simplicity! A program that could be just as confused as I was after hearing the note F would be well on its way to modeling human music-listening.

SUR LE PONT D'AVIGNON

Whereas for the “Triplet Scale” I described my listening process one note at a time, for “Sur le pont d'Avignon” I will proceed by roughly a measure at a time to speed things along.



Figure 3.18: “Sur le pont d’Avignon”, measure 1.

This melody seems even simpler, so far, than the previous melody. Instead of C–D–E, we have heard C–C–C. The repetition is almost too obvious to mention, but the presence of exact repetition of pitch is quite salient; we will automatically notice that these notes are linked together. There is one difference between the notes, however: the third note is a half note. Of course, when we first hear the third C, it doesn’t sound any different from the others, but eventually we realize that it is sustained for a longer time than each of the first two notes. The longer duration of the third C makes it likely for us to hear a grouping boundary after it. Therefore, as a result of all the pitches being the same and the long duration after the third note, I hear these three notes as a group.



Figure 3.19: “Sur le pont d’Avignon”, measures 1–2.

The second measure has the same rhythm as measure 1, and it also consists of three notes at a single pitch. Even after I have heard just the first or second D, I notice the similarity to the first measure and form an analogy between them, so I expect the third note to be the half note D that does indeed occur.

At this point, I have heard an *analogy* between the two measures, and also heard each measure as forming a *group* consisting of three notes. I’m not sure if I have strong expectations for measure three, but if anything I would expect the pattern to continue on an

E. The second measure sounds like the first measure transposed up by one step; it is easy to imagine this pattern continuing up another step.

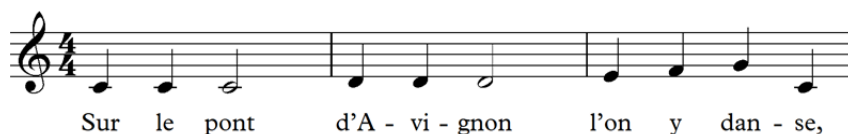


Figure 3.20: “Sur le pont d’Avignon”, measures 1–3.

The third measure starts out just as we might expect, with a quarter-note E. However, it diverges on the second beat and we hear a succession of different pitches; this is quite different from the pitch repetition of each of the first two measures. The notes E–F–G go together and sound like they form a group because they are a scale fragment. The next note, C, is a bit of a surprise, because it is the first non-stepwise motion in the melody. After an ascent all the way from the initial C in measure 1 to the G in measure 3, this note sounds like a sudden fall back down to the tonic, where the melody started. Because this fall lands on the tonic, I hear it as a conclusion to the group started on the note E. Thus, I hear all four of these notes as constituting a single group, although perhaps I hear the first three notes as a subgroup inside this group.

In any case, the notes of this third measure form a group. The rhythm and the pitch contour are quite different from those of the first or second measure, however. Therefore, when I hear measure 3, I also realize that the first two measures together form a unit, but measure 3 is starting something new. The groups in measures 1–2, then, seem to form a meta-group, and perhaps a new meta-group will start at the start of measure 3.



Figure 3.21: “Sur le pont d’Avignon”, measures 1–4.

Measure 4 sounds like measure 3; the similarity is obvious and jumps out, even without my consciously thinking about the precise intervals used in measure 4. Starting on the B, we hear another scale fragment going up, just as in measure 3. At some point in the measure, I notice the similarity, and perhaps I even expect the large leap down that happens after beat 3. (I can’t be sure about the details, because the first time I heard this melody was many years ago and now I know it very well, but it seems that early on in measure 4, any listener will hear the similarity of that measure to the start of the measure 3.)

At the end of the measure, the leap down to G is analogous to the leap down to C in the previous measure; measure 4 is an exact transposition (down a perfect fourth) of measure 3. Additionally, even if I wasn’t thinking consciously about harmonic implications in the first three measures, measure 4 clearly evokes dominant-chord harmony. The most salient notes in the measure are the B, D, and G — the notes of a G-major triad. The C sounds like a passing tone between B and D and it occurs on a weak beat (beat 2). The G is also on a weak beat, but it is salient because the melody lands on it after leaping down a perfect fifth. Applying the same reasoning to measure 3, we see that it is also easy to hear as emphasizing a C-major chord (the notes E, G, and C are salient). Hearing measure 3 with C-major harmony and measure 4 with G-major harmony makes measure 4 sound very strongly like a half-cadence, especially in conjunction with the grouping structure we have heard so far. An analogy was heard between measures 1 and 2 and they also formed a meta-group; similarly, an analogy was heard between measures 3 and 4, and they seem to constitute a second meta-group. The half-cadence in measure 4 makes this measure sound like the end of a group. Furthermore,

the cadence serves to close off other potential higher levels of groups: I hear the entire first four measures as forming a single group, which is brought to a close at this half-cadence on a G-major chord.

There is a slightly subtler higher-level analogy here. I can't be sure that a typical human listener hears it in exactly this way, but I'll mention it for completeness. Measures 1 and 2 sound like two instances of a musical idea, related by transposition. Likewise, measures 3 and 4 sound like two copies of the same thing, also related by transposition. In the first case, the transposition was a simple movement up by a whole step. In the second case, the transposition was larger: a movement down by a perfect fourth. But these two measure pairs sound analogous to me: the relationship between measures 1 and 2 is somewhat related to the relationship between measures 3 and 4, since both pairs involve transpositions.

I think that the analogy is even stronger, however, because of a stepwise connection linking measure 1 to measure 2 and a very similar connection linking measure 3 to measure 4. At the end of measure 1, the melody moves *up* by one step (this is trivial because we have been thinking of measure 2 as a transposition) as the C goes up to D, while at the end of measure 3, the melody moves *down* by one step: the C moves down to B. This is a much stronger connection than simply hearing measure 4 as a transposition: the entire measure is indeed a transposition, but at the note level there is a stepwise connection. I think of this as a meta-analogy: measures 1 and 2 are connected in a very similar way to how measures 3 and 4 are. Also, there is a nice symmetry in motion between these pairs of measures: the melody moves up to go from measure 1 to 2, but it moves down to go from measure 3 to 4. The rhythm also contributes to this symmetry: the initial ascent from C to E in measures 1–3 is quite slow, involving two half notes, but measures 3–4 consist exclusively of quarter notes that reach the high point of G but then rapidly come back down to C — and then fall past C

all the way to a lower G. Metaphorically, it seems almost as if it took a lot of work for the melody to “climb” up to the mountain peak of G, but this climb was followed by an effortless fall back down to the starting point, crashing through the C to bounce off B, only to fall again all the way down to the low G, which acts as a safety net that finally stops the fall. (Of course, it is hard to articulate or even to know exactly how we hear this pattern, but the distinctive shape of measures 3 and 4 surely evokes some unique feeling that contrasts with how we hear measures 1 and 2, and I suspect the rising and falling motion in the shape causes most listeners to hear something vaguely akin to what I described.)

Let’s keep listening — the next figure gives the next three measures:



Figure 3.22: “Sur le pont d’Avignon”, measures 5–7.

After just a few notes in measure 5, it sounds like the melody is repeating from the start. We hear the three C notes with the quarter–quarter–half rhythm and it’s easy to remember that this is the same thing we heard at the very start of the melody. As the melody continues into measures 6 and 7, there is no surprise; we hear an exact repetition.

The same groups form in these measures as in the first ones, although it is perhaps a little easier to make sense of the melody now, the second time around (it was easy enough the first time, but now we really know what to expect). In a trivial way, we form analogies between these measures and the first measures: measure 5 is exactly like measure 1, measure 6 is the same as measure 2, and measure 7 is the same as measure 3.

There is one slight difference, though. We have already heard the first four measures as a group ending on a half-cadence. The current group of measures sounds like a repetition, but familiarity with tonal music suggests that we might reach a *full* cadence soon, even

though so far we have exact repetition. In cases such as this one, I expect a tonic chord to arrive soon, perhaps at the start of measure 8, so I am waiting to hear the note C. However, measure 7 already ends with a C. It falls on a weak beat, however, so I anticipate a stronger C (perhaps reached via stepwise motion, as happens quite often in strong cadences).



Figure 3.23: “Sur le pont d’Avignon”, complete.

In the final measure, this expectation is satisfied nicely. The first two notes are still members of the dominant chord, D and B (over an implied chord root G), but the third note is a solid half-note C, which was indeed reached by a step and which sounds much more conclusive than the C in measure 7.

After hearing this final C, the group structure is quite clear: I hear the final two measures as forming a group (just as I did measures 5–6), at a higher level I hear the final four measures, 5–8, as a group, and finally I hear the entire eight measures of the melody as one big group. Because of the near-repetition of the first four measures (yielding the final four measures), the implied dominant harmony in measure 4 followed by the tonic harmony and melodic closure in measure 8, the first four measures sound like an antecedent phrase and the last four sound like its consequent phrase. A strong analogy links measures 1–4 with measures 5–8.

All in all, nothing was particularly surprising here, aside from the elegant meta-analogy that we might hear linking the pair of measures 1–2 with the pair of measures 3–4. (Notice that although measure 8 was not a transposition of measure 7, it was still linked to measure 7 by step, so it was not completely unlike measure 4 and its connection to measure 3.) This little melody seems rather easy to understand, and it induces some straightforward expectations along the way that mostly come true. I am quite fond of this little tune, especially because of the way measures 3 and 4 are linked together and because of the way the nonstop quarter notes in those measures, with their quick fall from a high G all the way to a lower G, contrast with the slow climb in measures 1–3.

ON THE STREET WHERE YOU LIVE

The melody of “On the Street Where You Live” is much more sophisticated than the previous one. Listening to this melody is therefore quite a bit more complex, so I will focus on just a handful of the most interesting issues that arise.

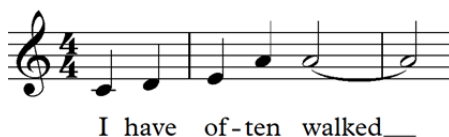


Figure 3.24: “On the Street Where You Live”, first 5 notes.

This melody starts out with the same three notes of the Triplet Scale melody, except that the first two notes are pickup notes: the E is on a downbeat instead of the C. Once the E has been heard, the first three notes will likely be heard as a cohesive group — an ascending-scale fragment in C major — just as they did in the melody earlier. At this point I might expect the scale to continue upwards to G or to descend back to C.

Therefore the next note, A, may come as a small surprise: instead of a continuation of the scale, there is a leap upwards by the interval of a fourth. This gap in pitch — the wide

interval — causes me to hear the A as rather separate from the first three notes. In a way, I hear the A as potentially forming the start of another group of notes — it naturally forms a group with the long A that immediately follows it, because they have the same pitch. However, I also hear the first A as a continuation of the initial motion, partly it feels unnatural to think of a group ending immediately after hearing the first downbeat of the melody. Therefore, for me the most natural hearing involves hearing the first five notes, C–D–E–A–A, as one a single group. The group certainly ends after the second A, however, because of its quite long relative duration.

Also worth noting is the tension that I hear in the melody at this point. The upwards leap, coming right on the heels of a small climb from C to E, makes it feel like the melody is suspended up in the air, waiting to come back down. I feel tension as a listener at this point, because the melody is at an unstable place. Not only is the pitch suddenly high, but also the note A, in C major, is tonally less stable than the strong C and E notes that came before.



Figure 3.25: “On the Street Where You Live”, first phrase.

Sure enough, in the next bit of music, the A resolves downwards, relieving the tension. The next notes, G–F–E, form a descending group. Because they are moving with stepwise motion, I hear them as notes in a scale. I also notice (perhaps only semi-consciously) a similarity to the initial C–D–E scale fragment. This analogy, though simple-seeming, is subtle in several ways. The C–D–E group is ascending, whereas G–F–E is descending. The pattern of absolute distances in terms of half-steps is different: C–D–E is made up of the interval pattern “+2, +2” (ascending by two half-steps between each pair) while G–F–E

consists of the pattern “-2, -1” (the interval between F and E is smaller than the G-F interval). Nonetheless, we effortlessly hear these as similar patterns, because we can hear the notes in terms of their position within the major scale: C-D-E is heard as 1-2-3 and G-F-E as 5-4-3, making interval patterns of “+1, +1”, and “-1, -1”. Moreover, both of these segments involve three notes, which start “stably” (on C or G), then pass through a less stable note (D or F), and both end on the relatively stable note E.

At the end of measure 3, the note C is heard again; the melody leaps down from E to C, much as it leaped up from E to A earlier. Even though this leap involves a different interval and direction (down a third instead of up a fourth), I still hear it as a leap in the same direction as the previous notes were moving. The C is then repeated (just as the A was) and I hear this second set of five notes as another group; the C, along with all the downward motion, has relieved the tension of the high A earlier, and the melody has reached a point of repose and closure.

This 5-note group, G-F-E-C-C, is analogous to the first one, in that they have very similar contour, as long as we substitute downwards motion for upwards motion. In addition, their rhythmic structure is identical: four quarter-notes followed by a whole note. Table 1 summarizes the similarities.

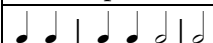

C-D-E (scale degrees 1-2-3)	G-F-E (scale degrees 5-4-3)
Ascending scale	Descending scale
Leap up (E → A)	Leap down (E → C)
Stable-unstable-stable	Stable-unstable-stable
A-A repetition	C-C repetition
	

Table 1: Similarities between the two halves of the first phrase of “On the Street Where You Live”.

A higher level of structure also emerges here. I hear a meta-group comprised of the first (ascending) 5-note group and the second (descending) 5 note group. This is quite similar

to the structure of the first four measures of “Sur le pont d’Avignon”: two measures ascend, and then two measures descend forming a symmetric large structure. In the present example, though, the analogy between the two halves of the meta-group is of a quite different nature from the analogy described for the previous melody.

So far, we have heard four measures worth of the melody. I’ll call this group a “phrase” for the purposes of this discussion. Let’s continue listening. I will only show successive phrases in the following figures because this melody is longer than the previous ones; a figure later in this section shows the melody as a whole.

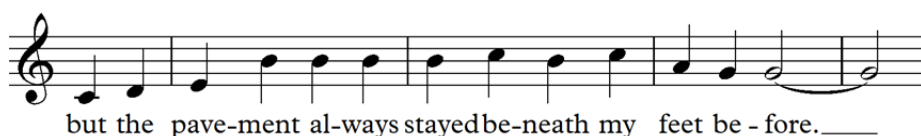


Figure 3.26: “On the Street Where You Live”, second phrase.

Since measure 5 begins with the same three notes as measure 1 (C–D–E), this might trigger a naïve expectation that the melody will continue in the same way as measures 1–4 did (after all, in “Sur le pont d’Avignon”, the first three measures did indeed repeat exactly). The note after the E, however, defies this expectation. That note, B, is indeed the result of an upwards leap from E, as expected, but the leap is larger this time: E goes up to B, not A. This note, B, sounds even more unstable than the A did, because B is the leading tone in C major.

The B is repeated, so that the first five notes heard are C–D–E–B–B. At this point, I expect the B to be sustained for four beats, just as the A was at the start of the melody. I am expecting to hear these five notes, ending with the long-duration B, as a group analogous the the C–D–E–A–A group. However, another surprise is in store: the B is only a quarter note long, and it is followed by another B. This is not too hard to understand, though: I expected the B to last for four beats, so I can hear this “extra” B simply as part of the expected long-

duration B, with an added rearticulation. Several identical notes in a row aren't terribly different from one long note, aside from short breaks in the sound and rhythmic emphasis occurring at extra points.

As I continue to listen, I hear yet another B — so far, there have been four B's in a row (instead of the expected two). But again, it sounds something like what I expected. Instead of a whole-note's duration worth of the B (after the initial quarter note B) I've heard three quarter-note B's. Maybe there is just one more left to go to fill out the duration of the whole note.

The next note, however, is a surprise: instead of another B, the melody ascends a half step more to the C an octave higher than where we started! (In the remainder of this section I denote this higher C with a prime symbol: C'.) Perhaps it is unsurprising that the leading tone resolved upward to C', but it increases the feeling of the melody having reached quite a high point, since after an initial ascent by step from C to E, followed by a large leap up from E to B, the melody is pushing even higher. Granted, the C' is the tonic and has some stability, but there is still a lot of tension here due to the height of the melody. (And although lyrics are not part of Musicat's domain, notice the text-painting here: the melody has soared up to C' just as the lyrics describe how "the pavement always stayed beneath my feet before". The "pavement" of the lower C is an entire octave below the current lofty height of the melody.)

A particularly interesting phenomenon here is the way in which this phrase is heard in relation to the first phrase. It is clear that I hear an analogy between the first phrase and this one, and the analogy informs my expectations about what is about to happen (such as my expectation for the B to be a long note lasting four beats). But even more interestingly, the current push up to C' starts to strain the analogy: the disparities between the two phrases are growing as the melody continues, yet there is still an obvious relationship between them.

The feeling I have at this point is similar to what I felt in the Triplet Scale, when I had to reinterpret what I heard as being in triple meter instead of 4/4. It is a bit confusing and the melody is not proceeding in the way I expected.

It's not only the analogy that is being strained or stretched here: the group starting with C–D–E–B–B is also being expanded to the right. I had expected the long note B to be the end of the group, but now “extra” notes keep coming, and there is no obvious breakpoint, so I mentally keep adding these notes to the group. The group so far contains C–D–E–B–B–B–B–C'. When will the group end?

The next note is another B (we are now on the second syllable of the word “beneath”). By this point, the expected four-beat B should have ended, but the melody is pushing along, still in B territory. Maybe it's finally time for the tension to be resolved and for the melody to descend as it did in the first phrase.

But we have yet another surprise: it's right back up to C' yet again! I feel a bit uneasy here as a listener: the succession of B's and C's is getting overwhelming as I yearn even more for the tension to be released. There is tension from the leading tone B's and from the ever-expanding group in my mind (now it has been extended to C–D–E–B–B–B–B–C'–B–C'), and I'm still waiting for the expected descent to happen.

Finally, the melody leaps down to A, bypassing the B. It's a welcome relief. The grouping structure I hear here is a bit unclear; perhaps I hear a new group starting on the A, or perhaps it continues to expand the ever-growing group that contains all the B's and C's.

After the A, the melody continues right on to G on the word “before”, and the G repeats. Suddenly, it sounds like we've reached the end of the second phrase, because the G is a long note, just like the C in the first phrase was. The very end of this second phrase sounds like the end of the first phrase to me, even though the three-note descending scale was

skipped this time. Also, the phrase doesn't end on the "pavement" of C, but rather on the alternate, fairly stable, dominant base of G. Because G is the second most stable note in the C scale, it is easy to make the analogy: both phrases end with a descent to a stable note that is repeated.

Here I will take a moment to discuss the grouping structure of the melody so far. At this point, I have heard the music as one large 8-bar meta-group consisting of two 4-bar meta-groups, with numerous relationships linking the 4-bar groups to each other. The first 4-bar meta-group is heard as being made up of 2-bar groups that are related to each other, and each of these may be further divided into yet smaller pieces just two or three notes long. Note how the perceived structure is built up of ever-larger structures, where each larger structure is roughly twice the size of its highest-level components. The tendency of simple melodies to be constructed in this way, with groups often having lengths that are a power of two (2, 4, 8, 16, *etc.*) is known to the Musicat program and provides a helpful heuristic for how to hear groups.



Figure 3.27: "On the Street Where You Live", third phrase.

The doubling in size of structural components described above continues apace, starting with measure 9. Again there are three scale notes starting the measure, but they are D–E–F this time instead of C–D–E. Despite this small change in pitch, I effortlessly hear an analogy linking these three notes to the three at the start of the first phrase, or maybe to the notes at the start of the second phrase (or both). These notes are followed "as usual" by a leap, but this time it is an even larger leap (a major seventh) all the way to E', which is much higher than the "pavement". (More text painting happens here: the large leap up to E')

perfectly echoes the idea that “all at once am I several stories high”. It’s a major 10th high, to be exact.) Next comes the analogy-based expected repetition of this note, and, as in the first phrase, it repeats just once, with a whole-note duration. I hear an analogy between this part of the melody and the group at the start of the first phrase. The melody is suspended at the dizzying height of E’, and it sounds like this is the conclusion of another group: D–E–F–E’–E’.

Now I expect the melody to descend again to the “pavement” C, but instead it moves down just a bit to C’, an octave above the C. The end of the phrase is still recognizable and reminds me of the end of the first phrase, despite many differences. Both phrases end on a repeated tonic note (C) and the last note of each phrase has a long duration. So, I hear a strong analogy between this phrase and the first one.

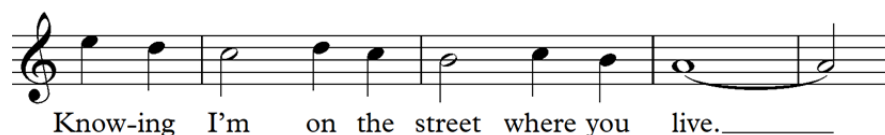


Figure 3.28: “On the Street Where You Live”, fourth phrase.

In the final phrase, a new rhythmic pattern begins. The first three phrases were quite similar to each other, except that in each successive phrase the high note was higher, and the second phrase had an interesting bunch of “extra” B notes. So where did this fourth phrase come from? It is very different from the first three phrases, but it doesn’t sound out of place.

Consider the first three notes of this phrase, E’–D’–C’. They seem to fit in well because they remind me of the notes that just came before. The same notes occur close to the end of the previous phrase, on the lyric “(sev)-ral stories high”. The rhythm is different in the third note, however: the C’ is a half note. This change in duration causes me to hear a group boundary as occurring after the C’; the notes E’–D’–C’ (“Knowing I’m”) form a small group.

At this point, several small groups and analogies form in succession. The next notes, D'–C'–B ("on the street") are a simple transposition of the previous three notes. The pattern continues with the next three notes, C'–B–A ("where you live"). This constitutes a musical *sequence*: The three-note seed, E'–D'–C', appeared three times, each time shifted down by a scale degree, and then the final A had a longer duration to finish the phrase. Naturally, I hear this entire sequence as a group, and there are straightforward analogies among all three components of the sequence.

7 I have of-ten walked down this street be- fore but the pave-ment al-ways

12 stayed be-neath my feet be- fore. All at once am I sev-ral

stor-ies high, Know-ing I'm on the street where you live.

Figure 3.29: "On the Street Where You Live", phrases 1–4.

The melody isn't complete yet: this excerpt ends on A, and eventually, of course, it will return all the way to the tonic and make a much more final conclusion to the song. The sequence ending on A yields only a temporary feeling of closure, but it does relieve much of the tension that was created by the high E' in the third phrase (as well as by the gradual climb up through A and B in the first two phrases).

There is much more to say about melody of the "On the Street Where You Live", but this description should illuminate some of the complexity of the listening process, and in turn, hint at aspects of listening that I intended to model with Musicat.

A Sneak Preview of Musicat

Chapters 5, 6, and 7 describe Musicat's listening in detail, for many songs, just as I have described my own listening in this chapter. But this seems an opportune moment to present a preview of what the program does. The final figure in this chapter shows the mental structures that Musicat formed after listening to “Sur le pont d’Avignon” on one particular run. The ellipses represent groups and meta-groups, while the thick parabolic arcs below the boxes represent analogies linking groups and linking meta-groups. (The thin arcs will be explained in Chapter 5.) Several (but not all) of the structures and connections I described above in my own listening experience were also “heard” by Musicat, and can be found in this diagram.

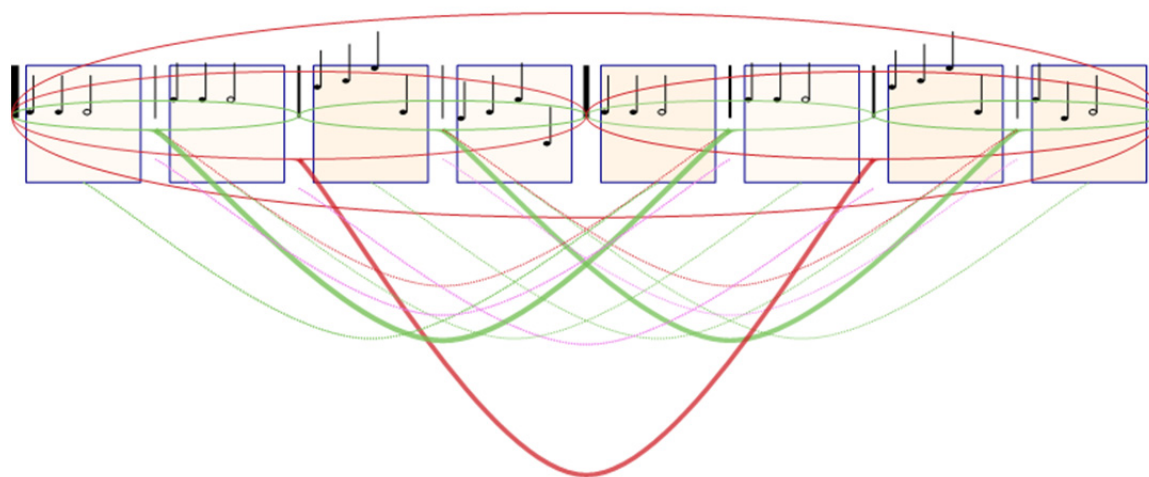


Figure 3.30: One of Musicat's listening performances for “Sur le pont d’Avignon”.

CHAPTER FOUR

Music and Analogy

Analogies All the Way Down

Douglas Hofstadter has argued that analogy is the core of cognition (Hofstadter, 2001; Hofstadter & Sanders, 2013, forthcoming). His view is that analogy-making is not a fancy cannon that we wheel out only on rare occasions to solve a tricky problem, but rather, that analogy-making happens all the time, involving everything from unconscious fleeting thoughts to conscious deliberation. Not only does making a statement such as “Iraq is another Vietnam” involve an analogy, but so does seeing a chair and mentally classifying it as a chair, or seeing the shapes **Z**, **Ƶ**, **ƶ**, and possibly **Ʒ**, and recognizing them as instances of the letter Z, or even my nearly unconscious deciding that this very clause should start with the word “or” while I was typing twenty words or so ago. According to Hofstadter, every word choice — even deciding between simple alternatives such as “and” or “or”, “the” or “a”, and so on — involves analogy-making. Analogies occur everywhere in thought and perception.

Given this viewpoint about analogy, it is natural to try to apply it to the particular case of analogy-making in music perception. We can imagine every bit of music perception as analogy-making, from recognizing a simple, primitive sound (a single note or a simple

sound in nature), to hearing a chord as either major or minor, to recognizing the repeating four-note “Fate” motif in Beethoven’s Fifth Symphony, to recognizing an entire performance as a reinterpretation of an older, well-known song, to hearing the style of a Chopin nocturne as similar to the style of a John Fields nocturne. In the following section, I will point out how analogy-making pervades the act of listening to Chopin’s “Raindrop” prelude, Op. 28 No. 15. The examples ascend through ever greater levels of complexity, from small-scale details to large-scale musical forms, and they thereby show how analogy is a fundamental component of music-listening at each hierarchical level. In most cases I will show examples of how two musical items are heard as “the same thing” even when there are real differences between the two items — the listener creates a musical analogy between the two chunks of sound and thus hears them as “the same”.

Ascending Through the Levels

To a music theorist, the note might seem to be the smallest unit of music and thus the logical starting point for an examination of analogy-making on many hierarchical levels. See, for example, Leonard Bernstein’s lectures comparing music to language, which start by likening notes to phonemes, motifs to words, and so on (Bernstein, 1976). However, an “atomic” note is not really the smallest unit after all. Instead, I begin by breaking the note into its constituent parts: the parameters that make up a note, such as its frequency, its timbre, and its perceived pitch, before ascending to higher levels of musical organization such as rhythmic structure, motifs, phrases, and sections.

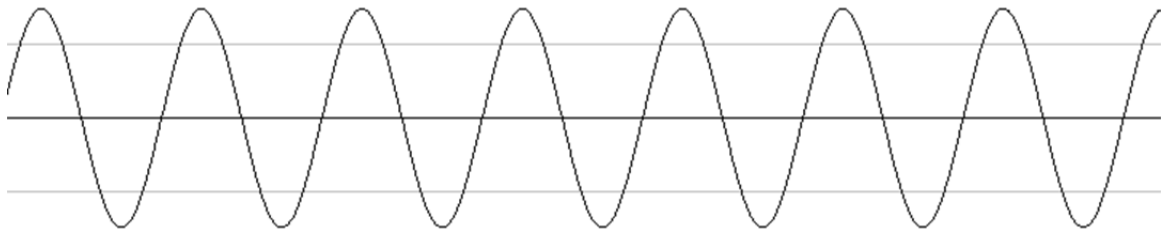


Figure 4.1: A pure tone at 440 Hz.

SOUND WAVES

The simplest possible musical tone is arguably a basic sine wave, as shown in Figure 4.1. The amplitude corresponds to air pressure in a sound wave, shown here with a frequency of 440 Hz (the note A in modern tuning). If the sound wave starts and then stops a second or two later, we can effortlessly hear this as a note, and even if we don't have perfect pitch part of our auditory system resonates with the particular frequency of the note. For such low-level perception, can we really claim that analogy-making is taking place? The physical structure of the ear causes fluid in the cochlea to vibrate in sympathy with the 440 Hz sound wave, and then hair cells in the inner ear corresponding to this frequency are stimulated, and the frequency and intensity of the sound are transmitted up towards the auditory cortex. One might think "Surely this isn't analogy-making — it's just a biological process." This is a reasonable position, but I claim that even this process of decoding a sound wave into a frequency (or set of simultaneous frequencies, in the case of a more complex timbre) is, in a tiny way, an example of analogy-making.

Analogy-making is simply the process of seeing something as similar to another, previously known thing. In the case of auditory processing, we hear a pitch (as opposed to a noise) when an extremely short-duration waveform is repeating periodically. In Figure 4.1, the sine wave sounds like a pure pitch at 440 Hz because the 1 complete period of the sine

wave shape (a peak followed by a valley) is repeating 440 times per second. Mechanisms in the ear, somewhat miraculously, can detect these 440 repetitions of the same “shape” each second. By detecting this repetition, a mechanical process in the ear is implicitly perceiving a sort of sameness: each of these patterns is like the others. Even though we don’t perceive repetitions or 440-ness in a normal cognitive sense, something inside us is responding to the sameness. This is a trivial and mechanical process, but we might still consider it an example of super low-level analogy-making (we might make the extreme statement that there are 440 analogies per second here, although compared with the much more sophisticated analogies people are capable of, perhaps we should call these micro-analogies or nano-analogies).

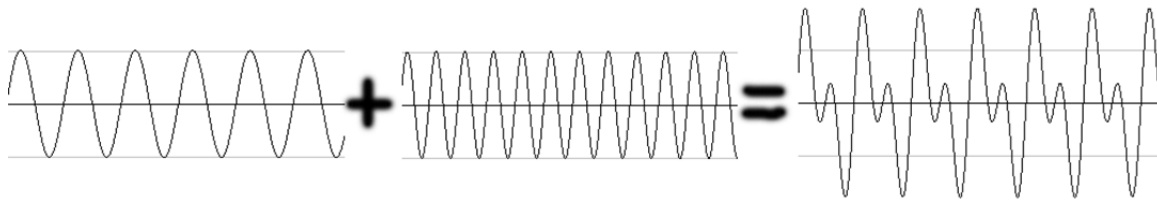


Figure 4.2: A composition of two sine waves at 440 and 880 Hz.

Lest the reader think that detecting sine-wave frequency is too trivial to be called even a nano-analogy, consider some more difficult tasks that our auditory system also does automatically. For instance, if we superimpose two sine waves of different frequencies we get a more complex picture (Figure 4.2), yet the ear is still able to detect both frequencies present in this more complicated shape.

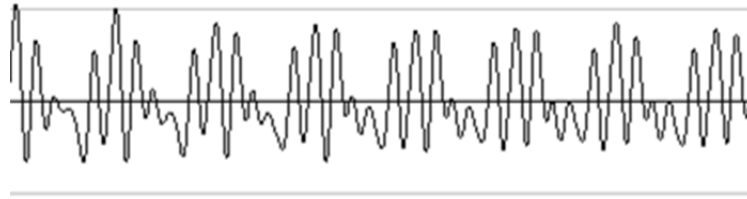


Figure 4.3: A small portion of the waveform of the note A played on a piano.

Finally, consider a more complex shape such as that produced by a piano playing a single note (Figure 4.3). The signal doesn't repeat exactly, but approximately, and the roughly-repeated shape is complicated, yet the brain is still able to detect the frequency of repetition and to make sense out of the complexity. The skeptical reader might point out that the ear is simply performing an operation analogous to the Fourier transform instead of really constructing an analogy. However, while the Fourier transform (a computational process for decomposing a signal into a sum of several simpler component signals) is a rather straightforward mathematical process, without the flexibility we associate with analogy-making in general, I like to think of this basic perceptual process as the lowest-level form of analogy-making (or nano-analogy-making) that we do when listening to music. However, even a single note can involve high-level analogy-making: we can recognize piano-ness and clarinetness and even Frank-Sinatraness in a single note. This is *high-level* analogy-making on a very low-level temporal scale. As we move upwards to consider larger levels of structure, higher-level analogies will be even more apparent and the analogy processes involved will become more complex and fluid. But these higher-level analogies rely on the foundation provided by our excellent pitch-detecting nano-analogy makers in the auditory system.

TIMBRE

Several years ago I had the sheet music to Chopin's "Raindrop" prelude sitting open on my desk, when my friend Alia saw it and asked, "Which piece is this?" I pointed to the

opening measure (Figure 4.4) and sang “Ya, da daaaaa, da daaaa... it’s Chopin’s Raindrop prelude.” She nodded in recognition.



Figure 4.4: The beginning of Chopin’s “Raindrop” prelude.

Naturally, even though I was singing the notes instead of playing them on the piano, my friend still recognized the melody. My singing created an implicit analogy between the sound of my voice and of a piano. Neither of us thought about it at the time, since such analogy-making is so natural, but it occurred nonetheless.

PITCH

One of the most important musical attributes of a note is its pitch. Still, it is easy to think of one pitch as being “the same as” another even when the frequency of the two sounds are different (excepting individuals with perfect pitch, of course). We are more sensitive to the position of a note in the tonal context of a scale than to its absolute pitch. The fundamental frequency of a note itself is not even so well-defined when we consider, for example, an opera singer with a large-amplitude vibrato.

The most basic example of analogy involving pitches is that we hear notes separated by octaves as very similar to each other. Octave equivalence is so directly based on acoustics that it almost seems like a property of the physics of sound instead of kind of analogy-

making. When a choir of men and women sings a passage in unison, for instance, it sounds like both groups are singing the same pitches, even though the men typically sing an octave lower.

When we notice that a pitch is repeated, we might consider this act of recognition to be analogy-making. One of the most characteristic features of the Raindrop Prelude is the nearly constant repetition of the $A\flat$ throughout. Each time one hears the $A\flat$, it is in fact slightly different from the others: the attack and loudness will inevitably differ from note to note, the previously sounding vibrations in the string will give the sound a different character, and the other notes present at the same time will affect the sound. Even more important is the harmonic context in which each $A\flat$ occurs. Indeed, although the frequency of the note does not alter one bit at the end of the first section of the piece, in measure 29 of the score it is replaced with its enharmonic equivalent, $G\sharp$, indicating that the tonal underpinnings have changed. The music modulates from $D\flat$ Major to $C\sharp$ Minor (Figure 4.5).



Figure 4.5: The key change to $C\sharp$ Minor between measures 28 and 29.

As soon as the bass chords begin at this key change, the $A\flat/G\sharp$ simultaneously sounds “the same” as before — it is, after all, a sound with the same frequency as before — but it also sounds darker in this new context. Indeed, as I listen to the measure before the key change, I anticipate the coming measure and notice that the $A\flat$ gradually takes on a more

sinister character. Paradoxically, the repeated note sounds like it is *remaining* the same (Ab) at the same time as it is *changing* into something else (G#). Here, analogy is a unifying force, keeping the perception of repetition intact to some degree even as the context changes.

RHYTHM

As soon as we contemplate musical structures larger than single notes, we encounter the concept of rhythm. Often, a characteristic rhythm becomes a unifying musical feature that suggests similarity between two otherwise disparate structures. A simple example is the incessant repetition of the eighth-note G# octaves from measure 36 through measure 40. The rhythm of repeated eighth notes is almost the simplest rhythm possible, yet in the next measure (measure 41) it becomes crucial to continuity as the most stable element of the piece suddenly changes. The G# octaves become B octaves (Figure 4.6).



Figure 4.6: G# becomes B in measure 41.

Were it not for the repetition of the eighth-note rhythm here, I might have written that the G# octaves are *replaced* by B octaves. However, the rhythmic pulse keeps going forward and the eighth notes keep repeating, making the analogy between the previous measures and measure 41 extremely clear. Even though this measure seems distinct from earlier material, in the sense that it feels like a climactic moment in the piece, it is nonetheless composed of the same sort of musical material. The minor chords of this dark

C#-minor section suddenly transform into a loud E-major and the unexpected B octaves appear, but we can hear a strong analogy to what came earlier, thanks in large part to the constant rhythmic pattern. In that sense, the G# octaves *become* (as opposed to *being replaced by*) B octaves,

The opening motif of the piece (Figure 4.4) also has a characteristic rhythm: dotted-eighth – sixteenth – half. The melody in measure 3 sounds very similar to that in measure 1 even though the pitches are quite different (except, of course, for the bass's repeating Ab). Rhythmic similarity also helps link the final measure here (measure 4) with measure 23. Both the initial phrase (measures 1–4) and a slightly modified version of this phrase (measures 19–23) end with a seven-note tuplet in the right hand. The second time it occurs, though, Chopin modifies the final seven pitches, and does not include the grace note at the end of the third beat that was present in measure 4 (Figure 4.7).



Figure 4.7: Measure 23 (analogous to measure 4).

“This phrase” appears again after the return to Ab-major near the end of the piece, and this time the tuplet has ten notes instead of seven, but it certainly sounds like more of the same thing. The only difference is that the flourish at the end of the phrase has become more elaborate, thanks to the extra notes (Figure 4.8).



Figure 4.8: Measure 85 (analogous to measure 23).

MOTIF

Continuing upwards to larger-scale structures, we encounter motifs — characteristic recurring short patterns of notes and/or rhythms. Familiar examples are found in Bach’s inventions, which are in large part based on the contrapuntal treatment of short motifs that return in all sorts of transpositions, inversions, and other modifications. Beethoven’s Fifth Symphony starts with the famous “Fate” motif and develops it throughout the rest of the first movement. Chopin may not make such obvious and pervasive use of short motifs as Bach and Beethoven in general, but the Raindrop prelude certainly does have one very significant and obvious motif: the repetition of $A\flat/G\sharp$. It is also possible to identify several other short motifs that gain significance during the course of this piece. I will examine some of these motifs before moving on to larger scale forms. During this analysis it is useful to keep in mind that motifs are highly relevant to the present discussion because when we notice a non-literal repetition of a motif, it indicates that we have formed an analogy.

$A\flat/G\sharp$ Revisited

I already mentioned the *fortissimo* moment where the repeated $G\sharp$ changes to a repeated B. Here, the rhythm and repetition of the motif make it clear that the B is still part of the same $G\sharp$ repetition motif, even though the pitch is different. This happens in a few

other places in the piece as well. Later in the minor middle section, G \sharp moves to F \sharp and then to A. The G \sharp switches between octaves a few times as well; sometimes it is lower, sometimes higher, and sometimes it is doubled. It even becomes C \sharp for a moment in measure 71 before the end of the minor section, as it moves from the upper to the lower octave. The A \flat in the first section of the piece also slips down to G \flat in measure 11, anticipating a descent through G \flat in measure 14 to become an F for five measures (Figure 4.9).



Figure 4.9: A \flat becomes F.

In all these cases, we still hear the repetition motif regardless of the difference in pitch. Even when it changes to F, it's still the "same thing" as the A \flat . The analogy is particularly strong here because the change to F occurs as the piece modulates to the key of B \flat -minor. F is the dominant pitch in B \flat , just as A \flat was the dominant of D \flat (and G \sharp is the dominant of C \sharp). The motif thus seems to be of the form "repeated eighth note on the dominant"; when perceived in this way, the analogy is very convincing.

Later in the piece, once the motif has been well established, even a single repetition of an eighth note in the proper context is enough to evoke the motif (Figure 4.10).



Figure 4.10: Another variant of the repetition motif.

Dotted-Eighth – Sixteenth – Half

The very first three notes of the piece in the right hand present an important motif with a characteristic rhythm (Figure 4.11).



Figure 4.11: Rhythmic motif.

When the same rhythm appears in the third measure, the similarity is immediately noticeable, even though this version has more notes playing simultaneously and moves *upwards* by *scale steps* instead of moving *downwards* by *leaps* (Figure 4.12).



Figure 4.12: Rhythmic motif appearing in another form.

This rhythmic motif also appears at the end of other measures such as measure 11, although here it is preceded by three grace notes and is in a different metric position — the last beat instead of the first beat — and does not evoke the original motif as strongly (Figure 4.13).



Figure 4.13: Another appearance of the motif.

Perhaps the original version of the motif (that is, the motif including the original pitches, not just the rhythm) appears relatively rarely in order to maintain its distinctiveness whenever it makes an entrance. Each time the F–D–A \flat version is heard at the beginning of a measure, it is a clear signal that the opening melody has returned. This is particularly important at the end of the minor section, when this motif signals the end of the middle section of the piece and the return home to the major key.

High Descending Notes

The next most salient motif to me (aside from the ornamental tuplet flourish discussed earlier, in the Rhythm section) starts with a series of eighth notes descending from the high points of the phrases in the B-section of the first page of the piece (measures 9–19), for example, the G \flat –F–E \flat –D \flat eight notes in measure 9. The motif concludes each time with a return to a longer note that is a bit higher than the final eighth note. The first instance is in measure 10, starting on the high G \flat (Figure 4.14).



Figure 4.14: Measures 8-11: High-descending-notes motif.

The next measure (measure 11) is the second instance of the motif. It is several steps lower in pitch, moves through a minor scale, and starts with a leap of a third, but it still sounds like the same thing. This motif comes back in a different form at the climax of the short coda of this prelude. The first note in the motif becomes much longer and higher (two quarter notes tied together on a high B \flat in measure 87), while the others become quarter notes and involve a large leap to and from the note C in measure 88 (Figure 4.15).



Figure 4.15: Return of the high descent motif: a ray of sunlight or “a star in the jaws of the clouds”.

This is a quite significant moment in the piece, as it the only time that there is a lull in the incessant repetition of the eighth-note motif. If I think of the “Raindrop” subtitle of the prelude, this moment makes me think of a ray of sunlight. And that image, in turn, makes me think of a vivid image described by Victor Hugo in *Les Misérables*, “a star in the jaws of the clouds” (“une étoile dans les gueules des nuages”) (Hugo, 1862, 1987). The analogy-making doesn’t stop.

PHRASES

A note to the reader: this section, on musical phrases, and the next, on large-scale form, will be easier to understand if you have the sheet music for the “Raindrop” prelude handy, because I will not reproduce the score here. I recommend finding a copy to use in following along, although it’s not strictly necessary.

The opening part of the prelude divides nicely into phrases. The first four measures make up a phrase (call it **a**). The next four measures are quite similar, but the phrase cadences and comes to a pause instead of moving forward with the seven-note tuplet of **a**. Otherwise the phrases are exactly the same. The analogy between them is almost trivial, especially since the second is a truncation of the first. I'll label this phrase **a'**. Next, a contrasting phrase (**b**₁) begins, although it only has three measures, followed by (**b**₂) for four measures and (**b**₃) for another four measures. The relationships between these phrases are more complicated and hard to pin down, even though it is easy to hear them as closely related. Each of these phrases involves a combination of the descending high-note motif and another vaguer motif involving quarter notes (measures 12, 16, and 18). After these **b** phrases, the **a** phrase returns two more times. The first return is the same phrase as before except for a modification of the pitches in the seven-note tuplet, while the final **a'** is much like **a**, only shorter and with a more conclusive coda followed by a transition into the minor section.

My grouping of measures into phrases and pointing out that **b**₁, **b**₂, and **b**₃ are related is based on my noticing of similarities between motifs, rhythms, textures, etc. in the phrases. That is, making analogies between phrases is based on the analogy-making occurring at smaller scales. It is more complicated to describe precisely the relationships between the **b** phrases than between **a** and **a'** because of the large number of elements that make up their similarity.

LARGE-SCALE FORM

The phrases of the first section group naturally into larger structures ("periods") so we can describe the first section of the piece as **ABA'**. The minor middle section has a more complicated larger form, although **CCD** is a good approximation. Finally, the coda might be labeled **A''E** (a variant of the original two phrases followed by an ending). Just as the **b**

phrases involved more sophisticated analogies between many elements, the minor section of the piece (**CCD**) is quite complicated and would take a good deal of prose to describe well. However, in listening to the middle section, it is easy to hear relationships between the phrases.

Overall, the piece itself follows an **ABA'** structure. This structure is easy to spot, thanks to the key changes that divide the piece into three sections, and the final **A'** is quite clearly related to **A** due to near-literal repetition of the initial phrase. Perhaps the piece is easiest to describe at this top level, rather than at middle or low levels, because so many details have been abstracted away by lower-level analogies. The **ABA'** structure indicates that we have an initial section, a contrasting middle, and a final section related to the beginning. Noticing this contrast involves analogy-making while we group all the pieces of **A** together and notice how they are distinct from **B**, but it seems that the most interesting analogy-making in this example happens at lower levels.

RAINDROPS KEEP FALLING...

Although Chopin himself did not give this prelude the “Raindrop” subtitle, it is difficult for me to dissociate the piece from thoughts of rain since the subtitle is virtually always attached to the piece. Naturally, this evokes a large-scale analogy wherein the music is compared with falling rain. When I hear the piece, the dotted-eighth opening note evokes thoughts of a raindrop collecting on a tree branch or overhang of a roof. The following sixteenth sounds like the point at which the mass of collected water becomes too great and the droplet breaks free of surface tension and falls down to the half note $A\flat$. It continues to the octave below, splashing into a puddle of many $A\flat$ eighth notes: “plunk, plunk, plunk”.

This interpretation of the opening motif is of course my own idiosyncratic view based on my personal experience, while the incessant “plunk, plunk, plunk” of the repetition motif seems likely to be a more universal interpretation. I find it likely that because of the name that has been associated with the piece, most people will automatically hear this “plunk, plunk, plunk” image of raindrops falling. Additionally, the minor section in the middle evokes predictably gloomy, stormy images. It is remarkable that music can evoke such concrete images about the real world. Listening to this prelude leads me to form detailed analogies between notes in the piece, between various rhythmic elements, musical shapes, motifs, phrases, sections, and so forth. These elements of analogy work at various levels in the piece to make sense of the structure and unify it into a whole piece that makes sense even at a representational level. I can listen to the prelude and, without difficulty, hear it as a light rain shower followed by a more serious windy storm, followed by a lightening of the sky, a burst of sunlight, and a final few plunks of raindrops before the end of the storm.

However, although raindrop analogies are easy to make to go along with this prelude, it is also enjoyable to listen to the piece without rain in mind. Indeed, in the *Unanswered Question* lectures, Bernstein (1976) challenges his audience to listen to a performance of Beethoven’s “Pastorale” Symphony without thinking of the cartoon images from the Disney movie *Fantasia*, in which animated stories are invented to go with the music of several classical pieces including the “Pastorale”. It’s a difficult challenge! But if you can strip away the animations from *Fantasia* or the subtitle “Raindrop” from the Chopin prelude, you can *still* hear *emotional* content. The act of perceiving various structures, motifs, analogies, expectations, tensions, resolutions, and so forth evokes emotional responses on its own.

Analogy as the Core of Music Cognition

We have seen many examples of musical elements that can sound like other elements due to analogies between the elements. How, though, do these examples support the thesis that analogy is the core of music cognition? What is meant by this statement? Specifically, I claim that

- 1) we *make sense of music* by making analogies between musical elements to aid in grouping and understanding; and
- 2) we *appreciate music* by enjoying the similarities and differences that go into each analogy.

An entire musical piece is a large, complex structure, which is digestible only in terms of smaller chunks. These smaller chunks are formed by grouping sections, phrases, measures, or notes together by virtue of similarities making up analogies. The act of perceiving analogies within a piece (or between pieces, or even between musical events and real-world events) helps give meaning to music. Musical semantics is different from the semantics of natural language because in general it is not representational, but when we listen, simply perceiving that a new musical element is similar to something heard before makes the new element sound meaningful. For example, the return of the opening motif at the return of the major section at the end of the “Raindrop” prelude sounds significant and means something to the listener just because of the happy recognition of the dotted-eighth motif. Much of our enjoyment of music comes from the analogies we spontaneously make when listening. For instance, the recognition that the A \flat has turned into a sinister G \sharp is a wonderful moment where the induced affect is crucially based on the listener perceiving an analogy between the notes.

Analogy is central in music cognition because it occurs spontaneously at every level of musical listening, from individual sounds to the form of an entire piece and even further “upwards”, connecting music with the external world. Additionally, parallel analogy-making at various different scales facilitates the building-up of larger and larger compound structures of musical understanding.

A subtle difficulty with analogy in music-cognition research is that analogy-making is so automatic for human listeners that it is hard to recognize that analogies are being made at all. My purpose here has been to try to illustrate the prevalence of analogy-making in music-listening in hopes that future work in the field may be motivated by recognition of linkthe centrality of analogy.

Analogy as the Core of Musicat

THE IMPORTANCE OF ANALOGY IN MODELS OF MUSIC COGNITION

Musicat without Analogy

Initial attempts at building Musicat did not include analogy as a central notion; likewise, most computational models of music cognition have little or no notion of analogy. Without analogy, however, Musicat was a collection of mostly low-level perceptual rules, which failed to make sense of musical structure. Even for structures as small as a musical phrase of four or eight measures, it was hard to make the program understand the grouping structure without reference to analogy. Once analogy was added, however, the program worked much better. Chapter 9 discusses this in more detail.

GTMM and Parallelism

In their seminal book *A Generative Theory of Tonal Music*, Lerdahl and Jackendoff do not use the term “analogy” *per se* but they do point out the importance of parallelism in musical grouping, which is a closely related notion. The relevant rule is “GPR 6”:

Grouping Preference Rule 6 (Parallelism) Where two or more segments of the music can be construed as parallel, they preferably form parallel parts of groups. (Lerdahl & Jackendoff, 1983, p. 51)

To “construe as parallel” is the heart of the matter here, and it is not obvious how to formalize this intuitive idea. Lerdahl and Jackendoff give a three-page digression (pp. 52–55) on the importance of parallelism, beginning with this passage:

The importance of parallelism in musical structure cannot be overestimated. The more parallelism one can detect, the more internally coherent an analysis becomes, and the less independent information must be processed and retained in hearing or remembering a piece. However, our formulation of GPR 6 still leaves a great deal to intuition in its use of the locution “parallel.”

Of particular interest is the authors’ humble admission that the details of parallelism constitute an important lack in their theory:

It appears that a set of preference rules for parallelism must be developed, the most highly reinforced case of which is identity. But we are not prepared to go beyond this, and we feel that our failure to flesh out the notion of parallelism is a serious gap in our attempt to formulate a fully explicit theory of musical understanding.

I claim that Musicat begins to fill this gap. Musicat presents an explicit model of analogy-making between musical structures, and parallelism can be considered as a particular application of analogy to the understanding of musical structure. And not only is formalizing analogy-making important to music cognition, but also it is important to cognition in

general, if we agree with Hofstadter's claim that analogy is the core of cognition. Optimistically, I hope that my work (like all the models developed by FARG) will have applicability beyond its particular (micro)domain. Lerdahl and Jackendoff have similar sentiments:

The problem of parallelism, however, is not at all specific to music theory; it seems to be a special case of the much more general problem of how people recognize similarities of any sort — for example similarities among faces... the hope of developing a solution to the musical problem in terms of the preference-rule formalism suggests that such a formalism may be more widely applicable in psychological theory. (Lerdahl & Jackendoff, 1983, p. 53)

Of course, my approach uses the FARG architecture instead of preference rules, but I share Lerdahl and Jackendoff's desire to apply lessons from work in music to other fields. Modeling analogy-making in music is essential to music (and Musicat), but it is also relevant to cognition in general.

This chapter has presented some evidence for the importance of analogy-making in music-listening, and therefore in the modeling of music cognition. The following three chapters demonstrate how these ideas have been put to use: in them we will see many concrete examples of analogies (and other structures) generated by the program Musicat.

CHAPTER FIVE

Musicat Listens to Bad Melodies

This section of my dissertation (Chapters 5, 6, and 7) shows Musicat in action as it “listens” to various melodies. Recall from Chapter 1 the concept of “listening as performance”. If this were a thesis on, say, a computer program that attempted to *play* Chopin nocturnes on a digital piano, this chapter would discuss the computer’s “performance” in the most usual sense of the word; that is, it would mean using an instrument to make music for an audience. However, this is a thesis on Musicat, and Musicat’s “performances” do not involve playing an instrument, but instead the dynamic act of *listening*. Listening is a temporal process that is hard to depict in static figures, and watching the program run is the ideal way to understand Musicat’s *listening performance* and to see how the program’s activity of listening, far from being passive, involves much work.

These three chapters give many examples of Musicat in the process of listening, using screen captures, or screenshots, to show Musicat’s internal representations of the music it is listening to. Early in this chapter I use many screenshots to give a crude approximation to a watching a movie of the program running. These “movies” are crude indeed — they show just one frame for each measure of music, whereas Musicat’s internal state may change not once but one *thousand* times per measure. Still, these crude “movies” should help the reader

understand how Musicat listens and how its internal representations (groups, analogies, and so forth) change over time as new melody notes are “heard” by the program. Later in this and followings chapters, to be more concise, I show nothing but the final state of the program after a run, even though this leaves out many important details of Musicat’s performances.

In this chapter I first explain how to interpret the figures generated by Musicat, and give a detailed example of its listening performance for the simple melody “Twinkle, Twinkle, Little Star” (this melody is re-examined in Chapter 6). Next, I show Musicat listening to melodies from three different categories: *Bad Melodies* (this chapter), *Simple Melodies* (Chapter 6), and *Complex Melodies* (Chapter 7).

The first category, *Bad Melodies*, is comprised of melodies composed on purpose (or generated randomly) to sound very “unmelodic” and to lack any sense of large-scale structure. Why should we subject Musicat to bad melodies? For one thing, the bad melodies are intended to act as a foil to the idea that a trivial program would perform just as well as Musicat. We show that Musicat behaves exceedingly differently depending on whether or not a melody has an interesting structure.

Consider the following source of concern: Musicat is biased towards hearing music as composed of regular binary structures; that is, it has a default expectation that the first two measures will form a group, as will measures 3–4, and then those two groups will be grouped together, and then the next four measures will show the same pattern, and this group will then join with the first four measures, resulting in a larger eight-measure group, and so forth, in ever-increasing powers of two. Analogies between groups, similarly, often show a regular structure where the left-hand side of each group is mapped onto the right-hand side of the group. One often (but certainly not always!) hears this sort of binary structure in simple melodies. One might wonder if a trivial program that understood nothing about music but

that *always* generated a regular binary structure of this type would perform just as well as Musicat.

All the Bad Melodies introduced in this chapter lack such structure. If Musicat were consistently to hear strong, regular, binary structures in these melodies, that would be most troubling. However, we show that Musicat is confused by the lack of structure in these melodies (as a human would be): it flails around trying to form coherent internal representations, and it generally fails to find strong groups and analogies.

The next category of melodies, *Simple Melodies*, consists of typical examples of melodies in Musicat's domain. They are rather simply-constructed melodies that should be easy for humans to make sense of. I investigate the types and strengths of structures that Musicat forms for these melodies, and show how its listening performances are different for these than for the bad melodies.

The final examples of this chapter are the *Complex Melodies*. These are not "complex" with respect to music in general, but they are certainly more complex than those in the Simple Melodies category. Complex melodies involve greater variation in structure and not as much literal repetition of melodic motives. They are also longer (that is, they may be 32 bars long instead of 8 or 16). We show how Musicat finds some interesting structure in these melodies, but also has more trouble understanding the complexity.

Understanding Musicat's Listening Performances

Musicat employs a unique graphical notation to display its internal representations of musical structure. Figure 1 shows an example, with labels identifying the different parts:

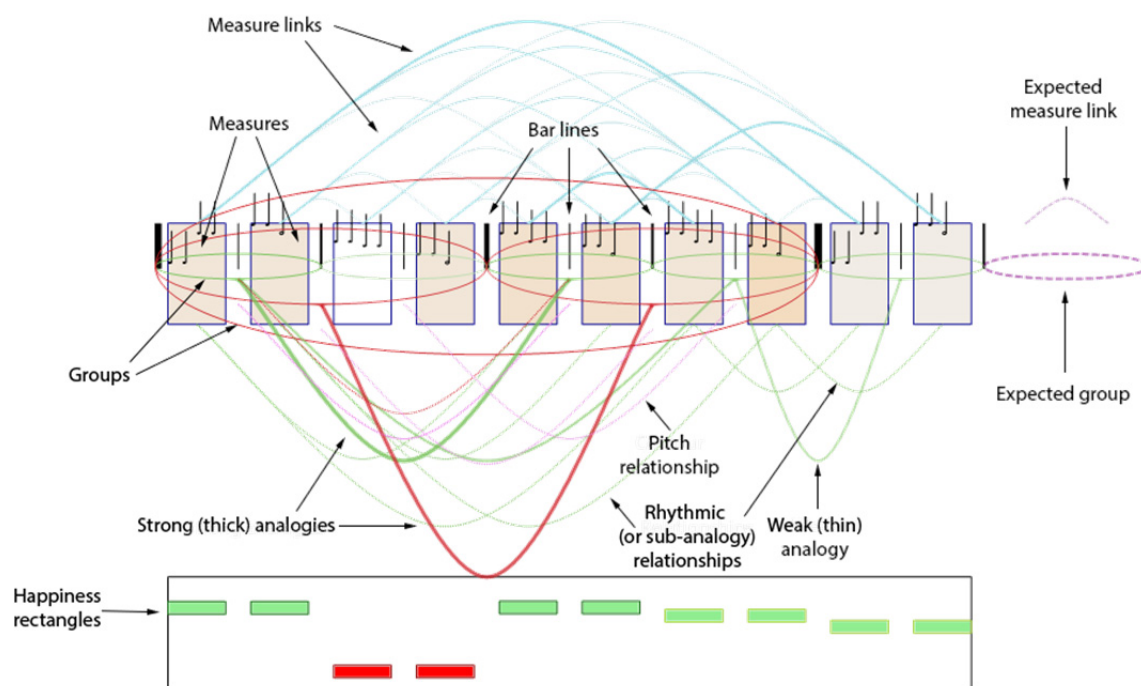


Figure 5.1: A sample screenshot demonstrating Musicat's notation.

The figure is divided into two major parts: music and associated structures on top, and a display of measure “happineses” on the bottom. Let's start with this simpler bottom part: it consists of a series of rectangles where the height (and, redundantly, the color) of each rectangle indicates the computed *happiness* of the measure directly above it. Happiness is described in more detail in Chapter 7. For now it is sufficient to understand that if the program has discovered good, strong structures associated with a measure, the measure is assigned high happiness; weak structures or missing structure (for instance, a measure that has not been assigned to a group) will have low happiness. In Musicat, happiness is inversely related to *temperature*, a notion borrowed from other FARG programs such as Copycat. The presence of temperature is most evident in the figure for high-temperature cases: low-happiness measures have high temperature, which is indicated by the red color of their

“happiness rectangles”, directly below them. (Low-temperature measures have green happiness rectangles below them, to emphasize high happiness, not low temperature.)

The top part of the figure displays the melody the program has heard so far, using a very simplified music notation, devoid even of staff lines. Each measure is represented by a rectangle containing the notes of the measure. Bar lines are drawn between the measures. Unlike in traditional music notation, these bar lines have variable thickness. A thick bar line indicates that the program expects the following measure to be heard as having a strong hypermetric accent, and thus that the bar line is a likely spot for a group boundary to form.

Each measure is shaded with a color corresponding to the amount of *attention* the program has given that measure in a very short recent time window. When the program is running, the measures flicker very noticeably as attention shifts between different measures.

Blue arcs are drawn at the top of the figure, above the music, connecting pairs of individual measures. These connections are called *rhythmic measure-links*, or simply *measure links*. They are formed when the program detects that two measures are similar to each other in terms of rhythm — pitch is ignored in the creation of these links. The strength of similarity is indicated by the thickness of the blue arc. In Figure 1, there are lots of arcs, which is unsurprising because each measure of this melody is one of only two possible rhythms — four quarter notes (QQQQ) or two quarters followed by a half note (QQH).

Measures can be grouped together. *Groups* are indicated by ovals enclosing (or superimposed upon) the measures in question. Groups may contain not only measures, but also other groups. Such groups (occasionally called *meta-groups* to avoid ambiguity) also are indicated by ovals enclosing the contained groups. The color of an oval varies with the size and strength of a group. The color also depends on context, and is chosen by the program to help make stronger and larger groups stand out. Note that there is no *a priori*, context-free meaning of group color. Typically, smaller groups are colored green and larger groups are

colored red, but as the program runs, a group that used to be colored red might turn green, or vice versa. This reflects nothing except for changing context. The thickness of an oval indicates group strength, as does opacity; for example, a weak group will be indicated with a thin and partly transparent oval.

The arcs drawn below the measures of music can be less easy to interpret because such arcs can mean several different things. The type of arc easiest to understand is a large arc attached on each end to a group: this is an *analogy*. For example, the large red arc in Figure 1 indicates an analogy between the group of measures 1–4 and the group of measures 5–9. Also notice that the groups linked by the analogy are both colored red. This is intentional: groups and analogy-arcs are chosen to be the same color whenever possible.

For each analogy, additional thin arcs are drawn using a muted version of the color used to draw the analogy. These arcs connect pairs of analogy components, in order to indicate the mapping that makes up the analogy; these arcs correspond to discovered *relationships* between components on either side of the analogy. Relationships can be divided into three categories: pitch relationships, rhythmic relationships, and analogy relationships. *Pitch relationships* (sometimes called “non-rhythmic relationships”) are based on pitch features such as contour or melodic tension. Pitch relationships are drawn with pink color (instead of the muted analogy-color) to distinguish them from the other two relationship types. *Rhythmic relationships*, the most commonly-appearing relationship type visible in the figures, indicate a relationship based on rhythmic features. *Analogy relationships* are indistinguishable in the figures from rhythmic relationships (when considering the drawing color alone), but they are not very important to the reader and I mention them here just for completeness: they indicate that an analogy has been chunked in order to be used as a component of another analogy. This is a bit redundant in the figure, because the original

analogy is also visible, but this type of relationship was necessary in the program for technical reasons. In the rest of this chapter and the following two chapters, for brevity, I discuss Musicat's relationships as if there were only two types: pitch and rhythmic; analogies are discussed separately.

Unfortunately, in the screenshots that follow it is hard to distinguish between pink arcs indicating pitch relationships and light-red arcs indicating rhythmic relationships for an analogy that is colored red. The pitch relationships were drawn with a finely-dotted texture in an attempt to distinguish them further, but this texture is also hard to discern. One clue is that most of the rhythmic relationships are drawn such that they connect with individual measure boxes, whereas the non-rhythmic relationships have endpoints that are near the groups involved and vertically positioned higher in the figure than the rhythmic relationships. I will comment on the type of these potentially ambiguous-looking relationships in the text when necessary.

Finally, at the right-hand side of the screen, we sometimes see structures indicating Musicat's *expectations*. Any expected structure is drawn using dashed purple lines. In Figure 1, the purple oval at the right indicates that the program expects a two-measure group to form in that location. Additionally, the purple arc above indicates that it expects a measure link to form between the two measures of that group.

SOME SIMPLIFIED NOTATION

For simplicity of notation, in the rest of this thesis, a group around measures 1–4 may be written with parentheses and in boldface as follows:

(1–4)

Even if a group such as this one contains subgroups, the group is still written as above, using just the measure numbers of the start and the end of the group; I generally avoid notation such as ((1–2)(3–4)) and prefer instead to leave the subgroups implied, to reduce clutter in the text. This is especially helpful in discussing analogies: using this notation I can write about an analogy between groups (1–4) and (5–8). Analogies also have a special notation and may be written with a double-sided arrow between groups:

(1–4) ↔ (5–8)

When referring to a range of measures that may or may not involve a group, I simply write the range of measures without boldface or parentheses:

1–4

A NOTE ON THE PRONOUNS “I” AND “WE”

Most of this thesis is written in the first person singular as I (Eric) describe my work. However, the pronoun “I” would be inaccurate in many places because I actually developed *Musicat* in close collaboration with my advisor, Douglas Hofstadter, over the course of many years in Bloomington and six months in Paris (or, as he would say, in “Villefleury” and “Lutèce”). Thus, there are places in this text where I write “we” instead of “I” to emphasize ideas or observations that are a result of this collaboration. This is especially apparent in this chapter and the two following ones, in which I slip back and forth between describing my own observations of *Musicat* in action and observations that were more collaborative. Thus, “I” refers to Eric, “we” refers to both of us, and the pronoun “one” is used where necessary to

discuss a general, abstract person (often “we” would be used for this latter purpose, but here “we” refers to two specific people). However, there are also some places where I find it natural to involve you, the reader, in my observations, writing phrases such as “we see in the figure that such-and-such has happened”. It should be clear from context when I am using “we” in this manner.

FIRST EXAMPLE: TWINKLE, TWINKLE, LITTLE STAR

Before jumping into examples of Bad Melodies, I present an example of Musicat listening to the familiar tune “Twinkle, Twinkle, Little Star”, which will appear again in the next chapter (this is an example of the Simple Melody category, even though the rest of this chapter deals with Bad Melodies). Remember that Musicat “listens” to music presented not as an audio recording, but rather as a series of symbols such as would be found in sheet music. We can approximate what Musicat receives as input if we imagine a sheet of paper covering up the entirety of Figure 2 below (the “Twinkle, Twinkle” melody) and then allow this imagined paper to slide slowly to the right so that it uncovers the notes one at a time, with the speed of sliding corresponding to the duration of each note.



Figure 5.2: Twinkle, Twinkle, Little Star.

Human listeners who already know this melody might remember the lyrics as well. In the rest of this chapter I present melodies along with their lyrics, simply as a memory aid for the reader — words make it easier to recall tunes. But keep in mind that the *program* has no knowledge of the lyrics to any of these songs.

Without further ado, let us watch Musicat “listening” to “Twinkle, Twinkle”.



Figure 5.3: Twinkle, Twinkle, with lyrics.

To start the listening session, the user simply starts the Musicat program, selects a melody from a list (or types the melody in using a special notation to indicate the notes and rhythms), and presses “Go”. The notes of the melody (“Twinkle, Twinkle”, in this example), start appearing on the screen and the program starts “listening”. The rate at which successive notes appear on the screen is roughly proportional to the musical duration of each note (the ratio being determined by the program’s current speed setting). For example, the first seven notes of “Twinkle, Twinkle” will appear with approximately the same amount of time in between each successive pair of notes. However, the seventh note (G, on “star”) is a half note, so if one second was spent on each quarter note earlier, it will take two more seconds until the eighth note (F, on “how”) appears.

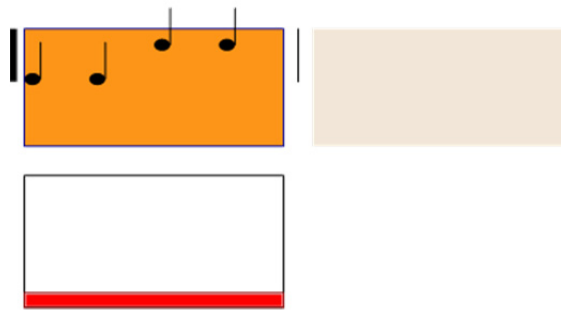


Figure 5.4: Twinkle, Twinkle, measure 1.

The figure above shows Musicat's display after the first complete measure has been presented. Because this version of Musicat is restricted to forming groups at least one measure long (and with a length that is an integral number of measures), no groups have formed inside this measure. (Otherwise, we would expect each pair of repeated notes to form a small group.) No groups or analogies have formed, but some other things have happened. A *thick* bar line has been drawn before the measure, to indicate that this measure has been heard as having a strong hypermetric downbeat. A *thin* bar line has been drawn after the measure, indicating Musicat's guess that measure 2 will probably be heard as being in a weak hypermetric position. The bar lines, incidentally, are drawn only in the top half of the space between measures, simply in order to de-clutter the display. The height does not represent anything, just as with a normal bar line.

The first measure has been shaded dark orange, to show that Musicat has focused nearly all of its attention on this measure. A small amount of the program's attention has been devoted to considering expectations about the second measure, as is indicated by the lighter-colored shaded rectangle to the right, in the spot where measure 2 will occur. The happiness of the first measure has been computed, resulting in the lowest value possible, because the measure is not part of any group.

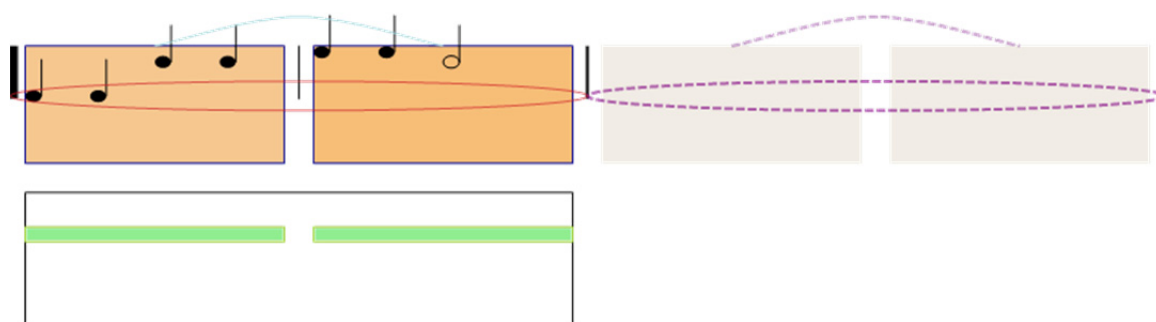
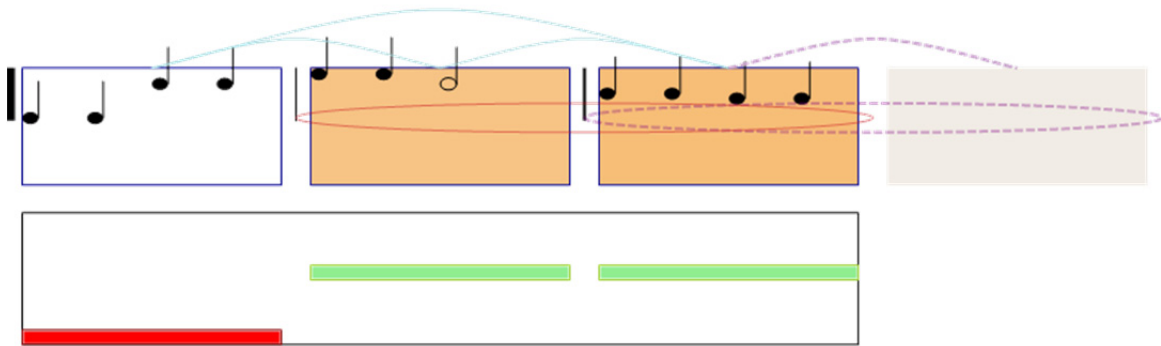


Figure 5.5: Twinkle, Twinkle, measure 2.

After measure 2 has been heard, several important things have happened: a link has been found between these two measures, because they have a similar rhythm. A group has also formed around measures 1–2, and an expectation has formed for another 2-measure group to follow. The first two measures also have relatively high happiness scores. Finally, we can see a bar line after measure 2 that is slightly thicker than the one preceding that measure.

Musicat's expectations (in the present version of the program) are not the melodic expectations for particular *pitches* to occur next, as in Larson's or Margulis's computer models; instead, they are expectations for particular *structures* to occur next. As will become obvious, Musicat's expectations are rather simple-minded; ideally, structural and pitch-based expectations would both occur in the model and would reinforce each other.

One important feature of Musicat's expectations is not shown explicitly in the figures, simply in order to de-clutter the display: when Musicat draws a purple expectation, it also creates an expectation (hidden in the figures) for an analogy to form later, linking the expected group to the group from which it derives. In the present example, Musicat expects another 2-measure group, (3–4), to form later on, and also expects an analogy to form between the red group and the future group (3–4).

Figure 5.6: *Twinkle, Twinkle*, measure 3.

After measure three, the picture looks confusing. Group (1–2) has unfortunately disappeared, and instead measures 2–3 have (also unfortunately) been grouped together. This is unfortunate, because we want Musicat to quickly and easily hear measures 1–2 as a group, and to not make the mistake of hearing such an unlikely-seeming pair as measures 2–3 as a group (for a human listener, the half note G in measure 2 is enough, in this context, to make the grouping structure quite obvious.) However, even though the creation of group (2–3) is disappointing for the human outside observer at the instant in time captured by the figure, it is nevertheless an expected and necessary consequence of Musicat’s architecture. The program changes its mind all the time, and any appearance or disappearance of a group, good or bad, is subject to reversal. Even though it looks as if just one group has been destroyed and another created (when we look at the difference between this figure and the previous), in reality many different group creation and destruction events might have occurred between the times that these snapshots were taken. Furthermore, the program’s creation and destruction of groups and other structures occurs in totally different ways on different runs, because of its nondeterministic or stochastic architecture. Thus, one should not be too surprised when groups such as (2–3) are formed — indeed, in some contexts such flexibility is absolutely necessary — but one should still hope that Musicat will quickly reverse any unnatural-looking decisions and settle on a human-like hearing.

Several other things have changed in the figure, compared with the previous one. Additional measure links have formed between all pairs of measures. The happiness of measures 2 and 3 is moderate, but the happiness of measure 1 is now zero; the red bar indicates low happiness (and hence high temperature). The program has spent most of its energy examining measures 2 and 3 (as indicated by the orange shading), but the low happiness should help it direct its attention back to the first measure.

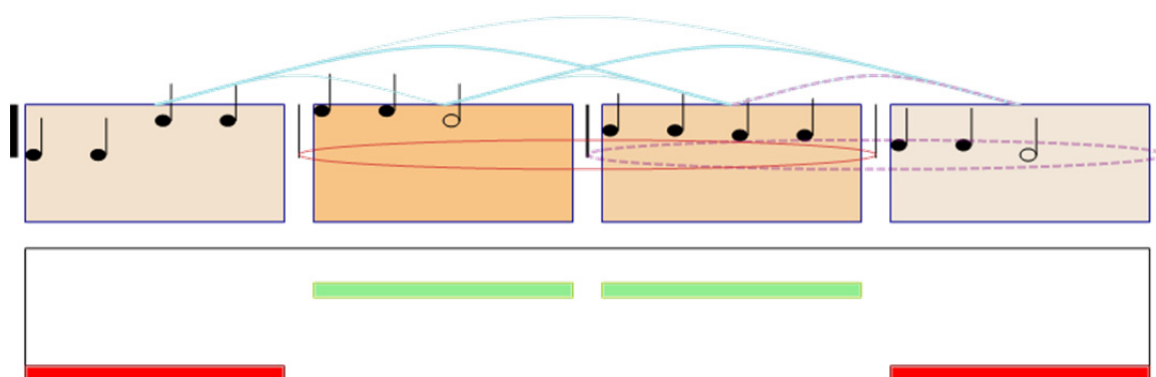


Figure 5.7: Twinkle, Twinkle, measure 4.

After measure 4 has arrived, the same strange group (2–3) persists, as does the expectation for the group (3–4). The happiness score for measure 4, as for measure 1, is zero. More measure links (based solely on rhythm) have been formed. In this picture we can see how measures 1 and 3 are strongly linked (after all, they have identical rhythms — four quarter notes in each), as are measures 2 and 4, also because their rhythms match. Another thin bar line has formed after measure 3.

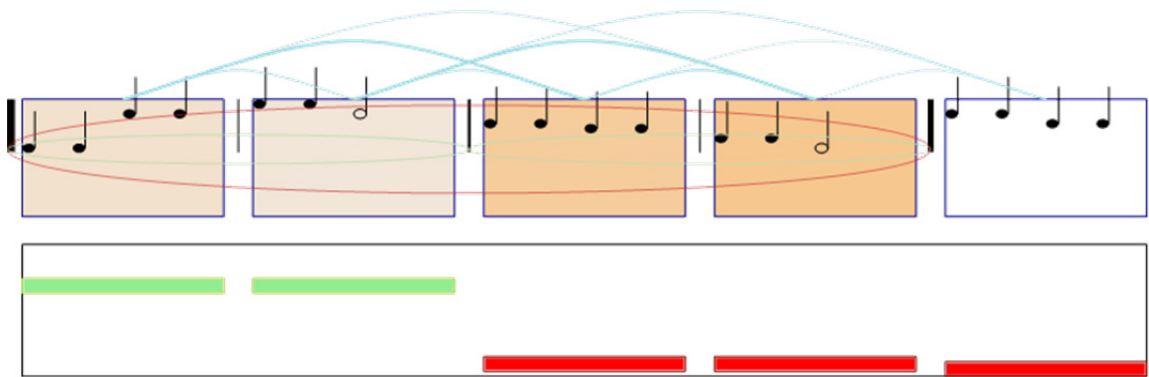


Figure 5.8: Twinkle, Twinkle, measure 5.

Suddenly, after measure 5 has been heard, the grouping structure has changed radically. Measures 1–2 as well as 3–4 form (light blue) groups, although they are difficult to see in the figure — you may have to look hard to see them! A larger-scale meta-group (red) has formed around those two groups. The strange group (2–3) has disappeared (fortunately). A thick bar line has formed after measure 4. The first two measures have high happiness scores again, although the program is unhappy about the rest of the measures, because group (3–4) has been perceived as very weak and measure 5 is not involved in any group at all.

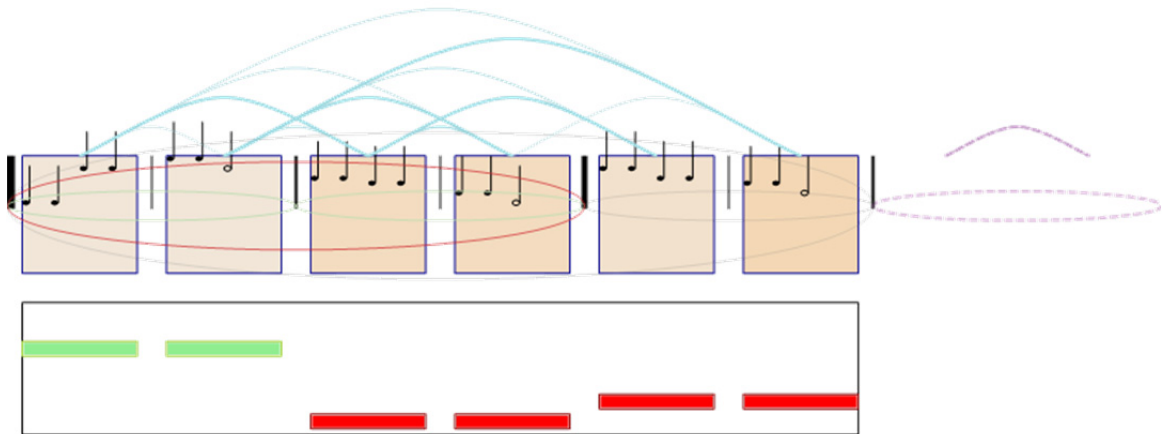


Figure 5.9: Twinkle, Twinkle, measure 6.

After measure 6 has been heard, an additional group has formed, (5–6), and it has been incorporated into a very weak (and thus barely-visible) meta-group, (1–6), encompassing all the measures heard so far. Another expectation has formed, this time for the 2-measure group (7–8).

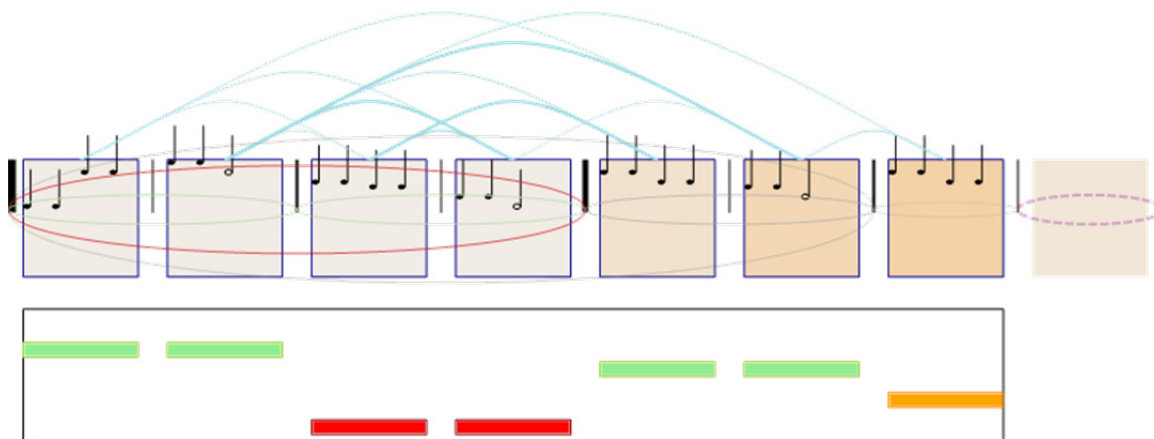


Figure 5.10: Twinkle, Twinkle, measure 7.

Here, after measure 7 has been heard, the structure formed looks similar to that in the preceding figure, although the happiness values for measures 5–6 have increased, and a weak 1-measure group has formed that contains the most recent measure. The observant reader may wonder why measures 3–4 have such low associated happiness values, while measures 5–6 have improved in happiness. The answer is, in part, that group (5–6) is now contained in group (1–6). However, this isn’t enough of an explanation, since group (3–4) is also contained in a parent group, (1–4). The full story is that as Musicat runs, it builds up a list of qualities of each group, and this list is used to compute group strength. (I call this the list of “Group Reasons” in a later chapter, but it refers to the same thing.) Like everything else in the program, this list is created in a stochastic manner, with different group qualities being noticed in different orders on each run. In this run, Musicat either has not noticed

some of the strong qualities of group (3–4), or else it has perceived some negative quality that makes it think it is a very weak group. While the program is running, I can pause it and display the list of group qualities, but in most cases I have let the program run without stopping to record these details. Calculation of group strength (and the associated measure happinesses) is an important topic, but this section of my thesis is focused on Musicat’s behavior instead of the lowest-level mechanisms underlying the behavior — see Chapter 9 for the details. In the present example, I would indeed like to see what happened to group (3–4), but I can’t simply re-run the program because the program is stochastic and I neglected to record the random seed used for the run; moreover, I made some slight changes in the code after this particular run, so now the behavior would be slightly different, even if I had access to the random seed.

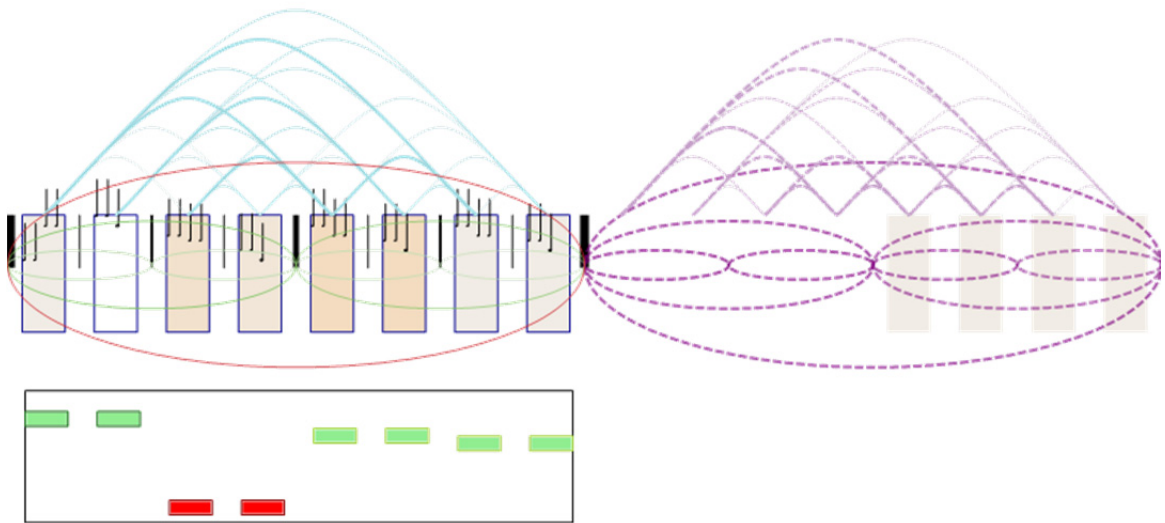


Figure 5.11: Twinkle, Twinkle, measure 8.

Now, after measure 8, another dramatic change has occurred: all of the first 8 measures have formed very regular duple structures. The program is happy with all the measures except for 3–4; for some reason, it still considers these to form a weak group, and these two measures will remain unhappy for the duration of the run. Additionally, a very

large and complex expectation has formed for another entire 8-measure structure like this one to follow (note that “Twinkle, Twinkle” only has 12 measures, so this expectation obviously will not be fulfilled).

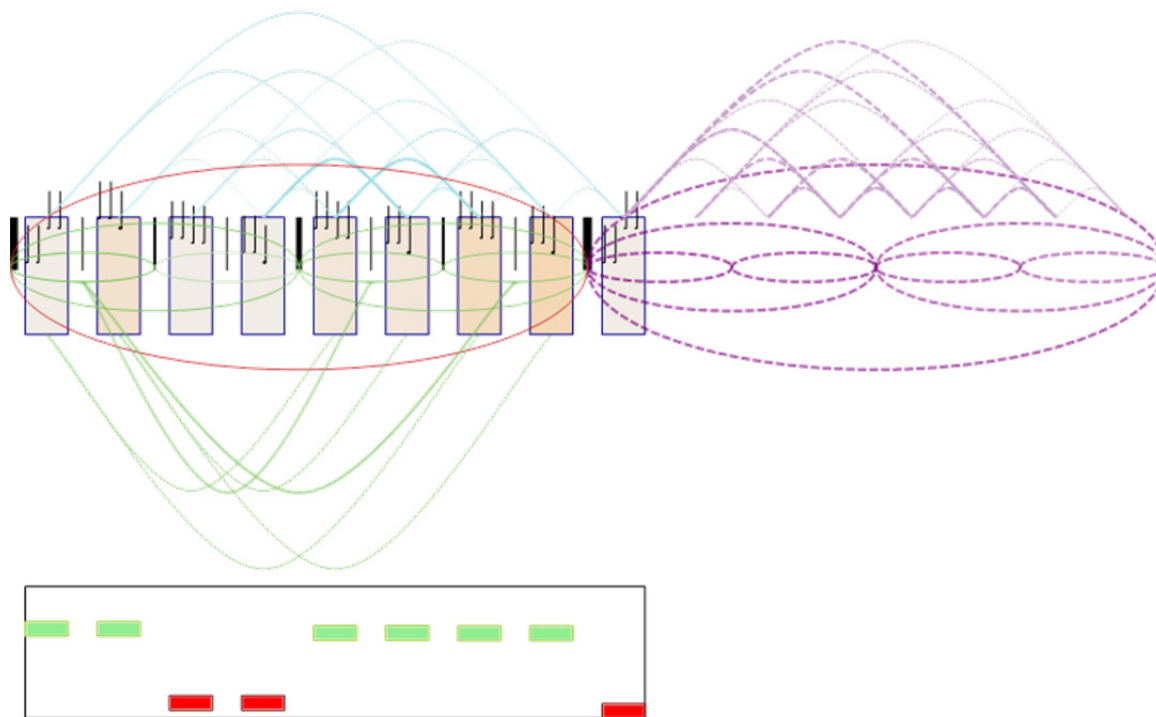


Figure 5.12: Twinkle, Twinkle, measure 9.

Suddenly, two analogies have formed! Both involve the group (1–2): one makes a link between (1–2) and (5–6), while the other makes a link between (1–2) and (7–8). It should be easy to see that measure 2 is similar in shape to measures 6 and 8, and also that all three of the groups involved, (1–2), (5–6), and (7–8), have the same rhythm (QQQQ QQH). Thus, these analogies should make intuitive sense. However, one obvious similarity in the music has not been noticed by the program: (5–6) and (7–8) are identical! The most obvious analogy in this melody is the trivial mapping (5–6) ↔ (7–8), but sadly Musicat has

not noticed it. This oversight is very weird, but don't lose hope yet — perhaps the analogy will be formed later in this run.

Not much has changed in terms of grouping. The new measure has been heard, but not incorporated in any structure, so it has low happiness.

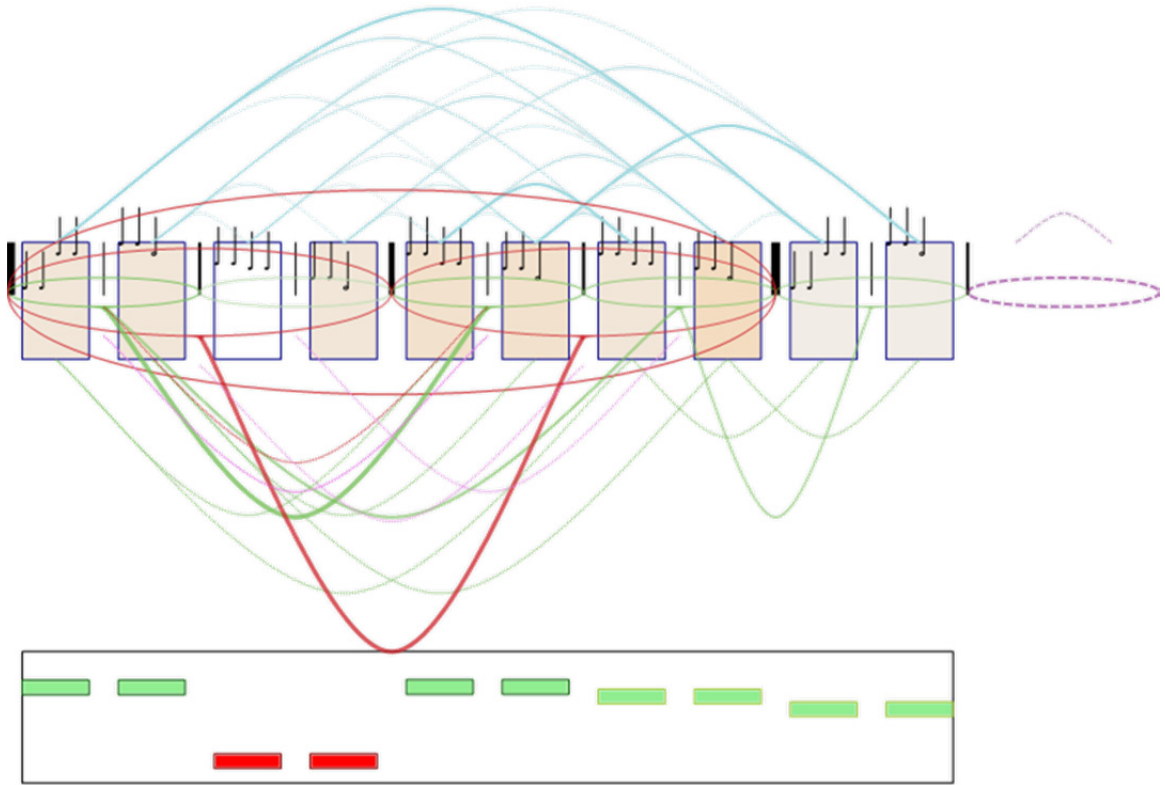


Figure 5.13: Twinkle, Twinkle, measure 10.

Now, just one measure after the first analogies appeared, a larger-scale analogy has formed between groups (1–4) and (5–8). These two groups are colored red, matching the color of the analogy, to make it easier to see the structure. The analogy between groups (1–2) and (5–6) has been strengthened.

The analogy between (5–6) and (7–8) is still missing, however. Likewise, human listeners will notice another exact repetition that has occurred: measures 9–10 are a

recapitulation of measure 1–2; they are exactly the same and occur in a similar hypermetric position, so it is trivial for a person to hear the analogy $(1-2) \leftrightarrow (9-10)$. Notice that Musicat has created a strong measure link (based on rhythm) between measures 1 and 9, and a weak one between measures 2 and 10, which should give it a hint to look for an analogy involving those measures. Likewise, there is a strong measure link between measures 5 and 7, so it has noticed part of the similarity required to make the $(5-6) \leftrightarrow (7-8)$ analogy. Will Musicat notice either of the missing analogies?

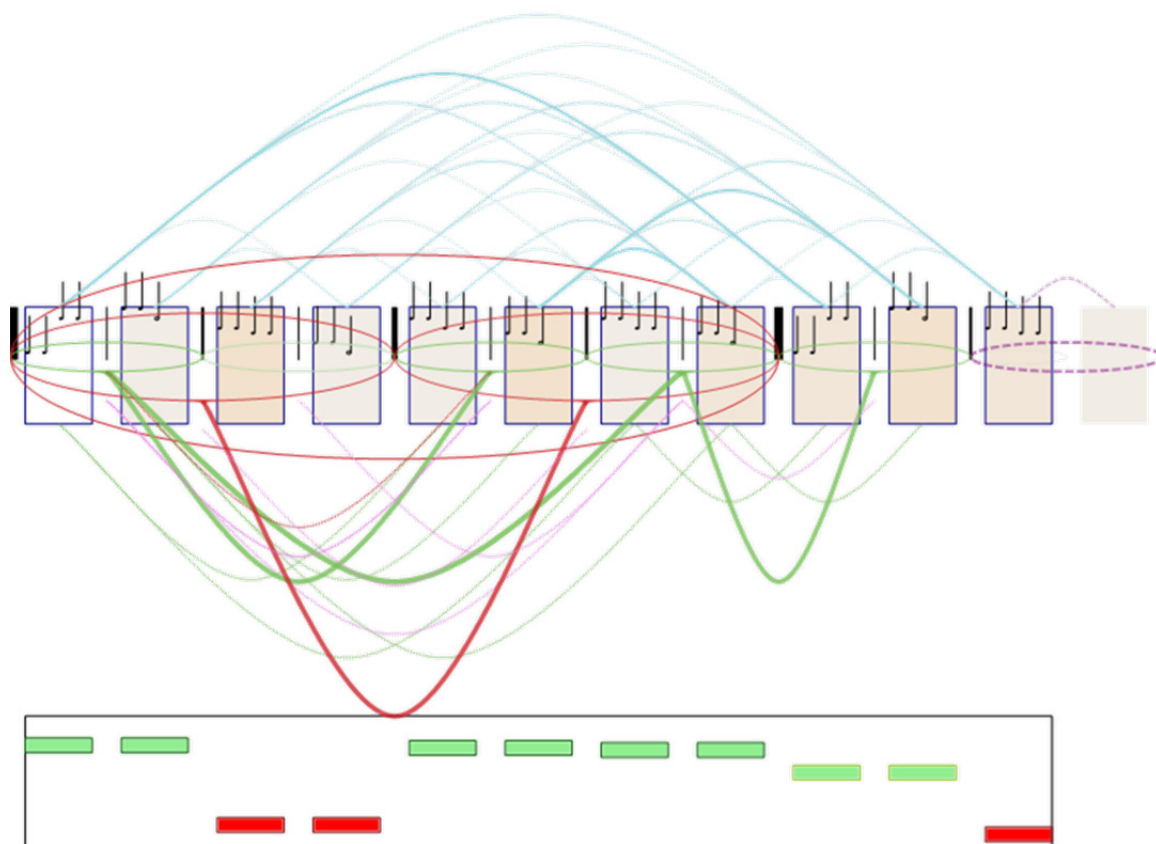


Figure 5.14: Twinkle, Twinkle, measure 11.

The same structures are in place as in the previous view, but some analogies have become stronger. Conversely, some of the measure links have weakened over time; notice how *recent* measures have long-distance links to earlier measures (such as the strong links above the measures linking measure 1 to 9 and measure 6 to 10), but there are not many links between pairs of *old* measures; those have mostly faded away.

Our two missing analogies are still missing, and the link between measures 5 and 7 has faded, so there is still some hope to see the (1–2) \leftrightarrow (9–10) analogy, but less for the other.

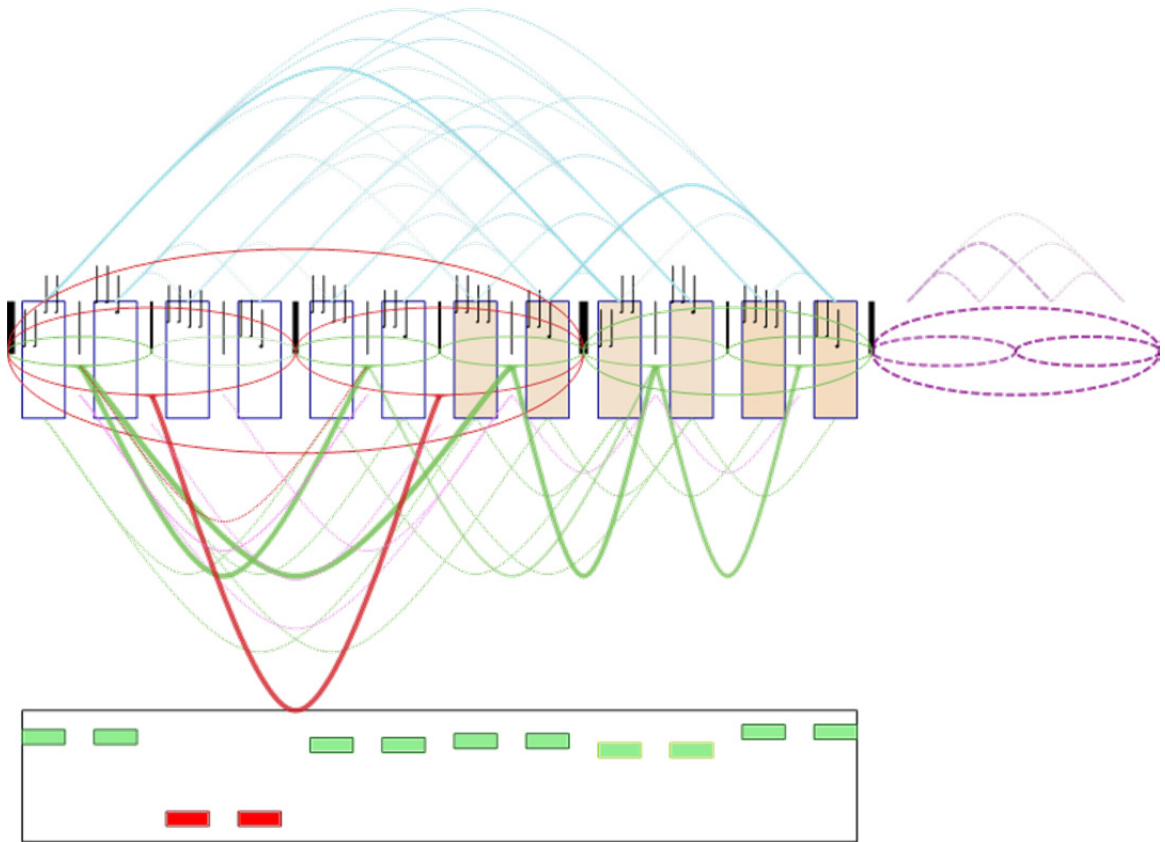


Figure 5.15: *Twinkle, Twinkle*, measure 12 (end of song).

The song has ended, but Musicat doesn't know it yet. The program has formed a new group, (9–12), and expects another 4-measure structure to occur. This is not surprising, because 16-bar songs are extremely common. A new analogy at the end of the melody, (9–10)↔(11–12), has been created.

The analogy (5–6)↔(7–8) still has not formed, and there is little reason to hope Musicat will notice it now, since its attention is focused on recent music. However, the blue measure link between measures 1 and 9 is still relatively strong, so there is still hope that the analogy (1–2)↔(9–10) will appear.

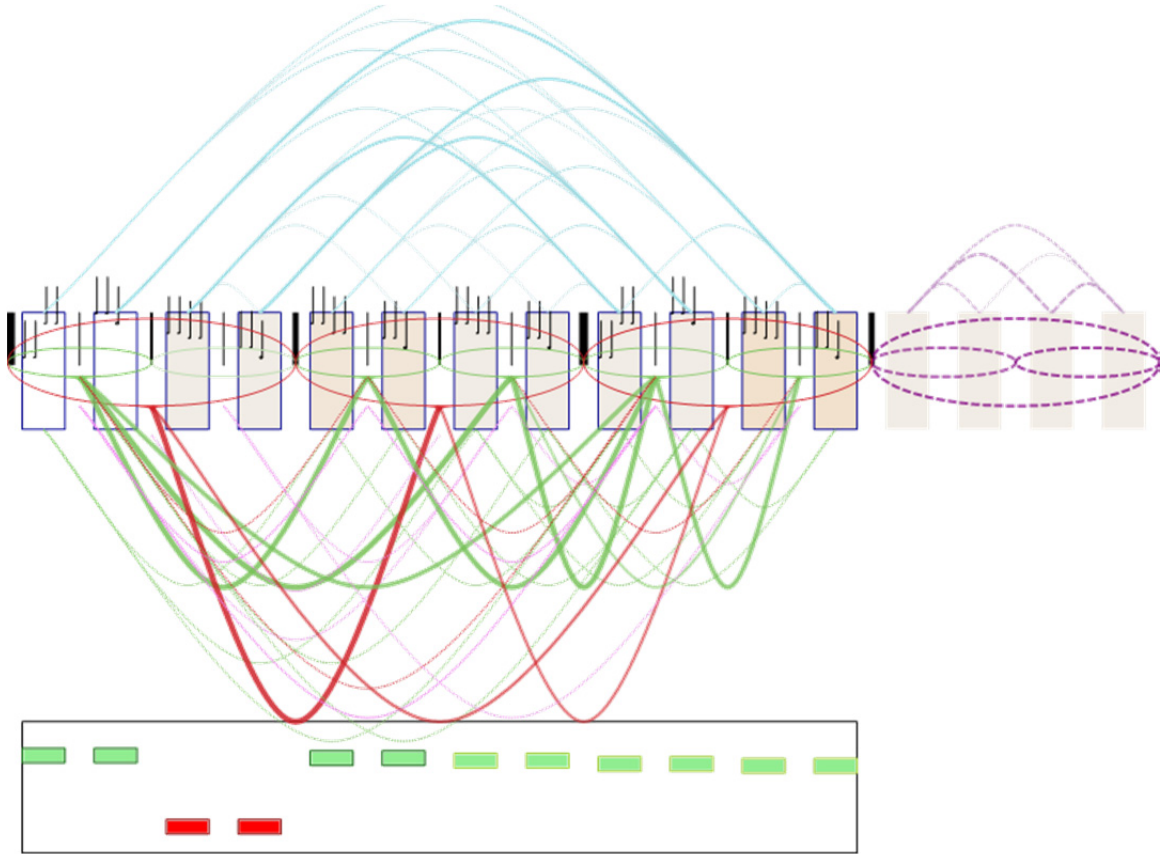


Figure 5.16: Twinkle, Twinkle, processing.

Processing continues for several measures' worth of time after the melody is complete. After some additional time, the analogy $(1-2) \leftrightarrow (9-10)$ has formed at last! This is important because it was such an obvious analogy for a human listener, and also because these low-level analogies help pave the way for larger ones. Indeed, two more high-level analogies have formed. The new (thin red) analogy $(5-8) \leftrightarrow (9-12)$ is weaker than the (thick red) analogy $(1-4) \leftrightarrow (5-8)$, but note that they are analogies each composed of essentially the same musical material, since the notes in measures 1-4 and 9-12 are identical. This similarity has been noticed by the program: a long-distance (thin red) analogy has formed between these two groups: $(1-4) \leftrightarrow (9-12)$, no doubt made possible by the $(1-2) \leftrightarrow (9-10)$ analogy.

All three of these large analogies seem to be reasonable; it is easy to hear similarity (or identity) in these pairs of groups.

In this large mass of analogies, it is disappointing that the analogy (5–6) \leftrightarrow (7–8) never formed; Musicat seems to have seen a great deal of the other interesting relationships in the melody. But each run is different, and in each run it pays attention to different features of the music.

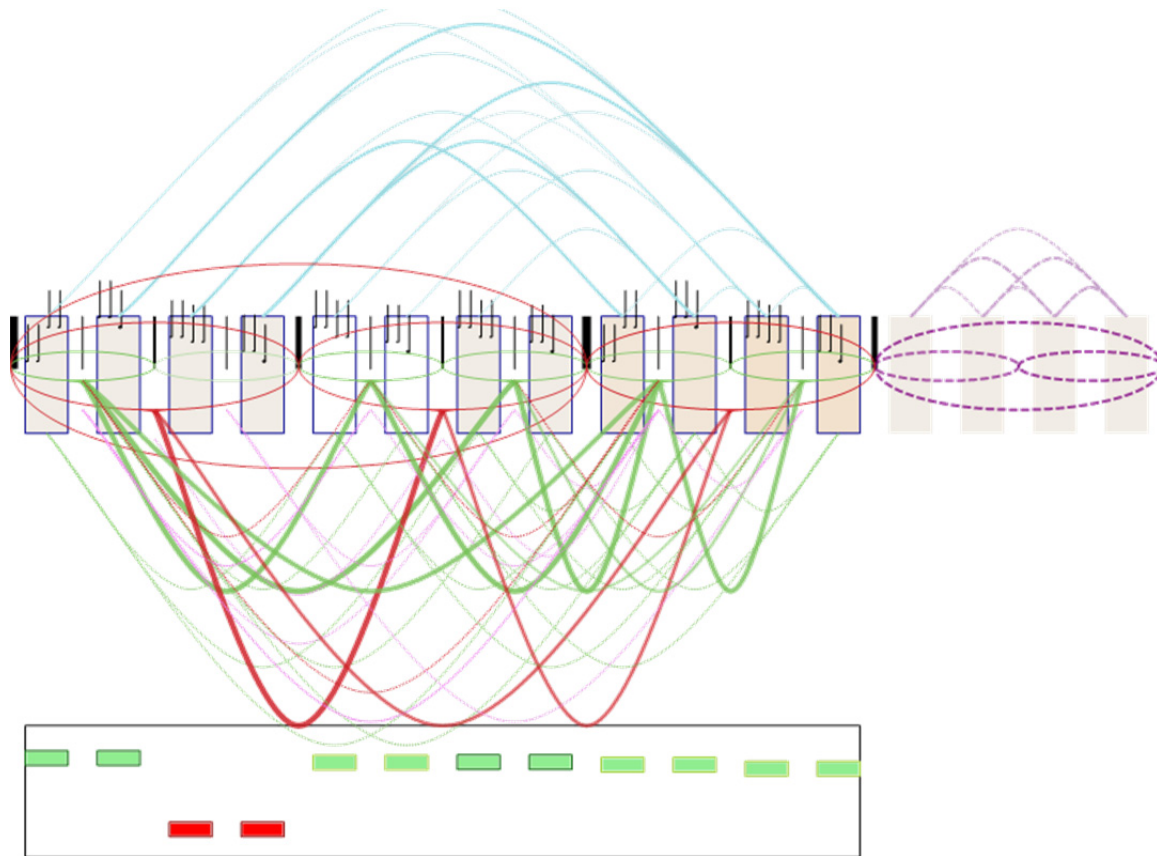


Figure 5.17: Twinkle, Twinkle, processing.

In the previous figure, Musicat is in its last moment of “listening” before the song is completely finished. The structures in the previous figure have been augmented with one more meta-group, (1–8). Most of the measures, except 3–4, have high happiness.

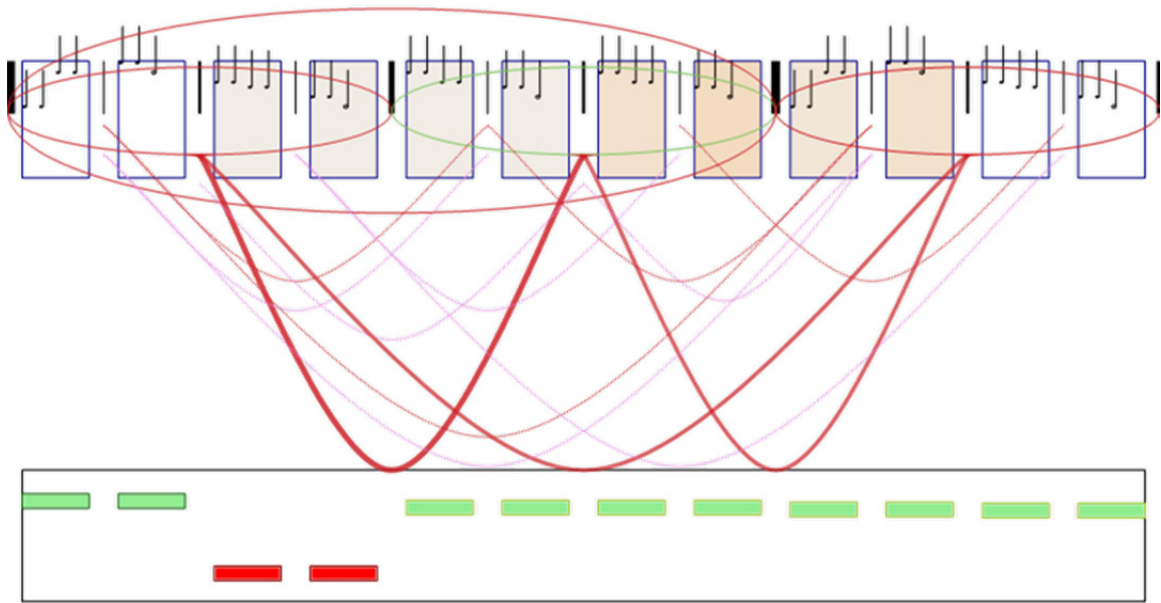


Figure 5.18: *Twinkle, Twinkle*, final state, low detail level (strongest structures only).

This figure shows the strongest elements remaining in the Workspace at the end of the run. The most important things are the three large-scale (red) analogies, the three 4-measure groups (1–4), (5–8), and (9–12), and the larger 8-measure group (1–8).

On this run, Musicat has heard the music in a fairly reasonable way. To human ears, this melody is an example of a simple **ABA** form; that is, we hear an opening 4-measure theme, a separate 4-measure long middle section, and then an exact repetition of the first theme. In this run, Musicat has noticed some of this structure; the relationship (1–4)↔(9–12) has been heard (although Musicat represents it as a strong *analogy*, not as a literal *repetition*). The middle section is in its own group (5–8). But the program has also

heard a higher-level 8-measure grouping, so “(AB)A” would be a better description of its listening than simply “ABA”. Additionally, it has heard myriad smaller connections between measures and groups. Interestingly, the analogies (1–4)↔(5–8) and (5–8)↔(9–12) indicate that the A and B sections are related; the mere description “ABA” fails to capture this relationship.

The most glaring problem in this run was the program’s “deafness” to the similarity between (5–6) and (7–8); these identical measures should have been heard as (trivially) analogous. But fortunately, on many other runs with the same melody it does notice this analogy; for example, see the next figure, showing the end of another run.

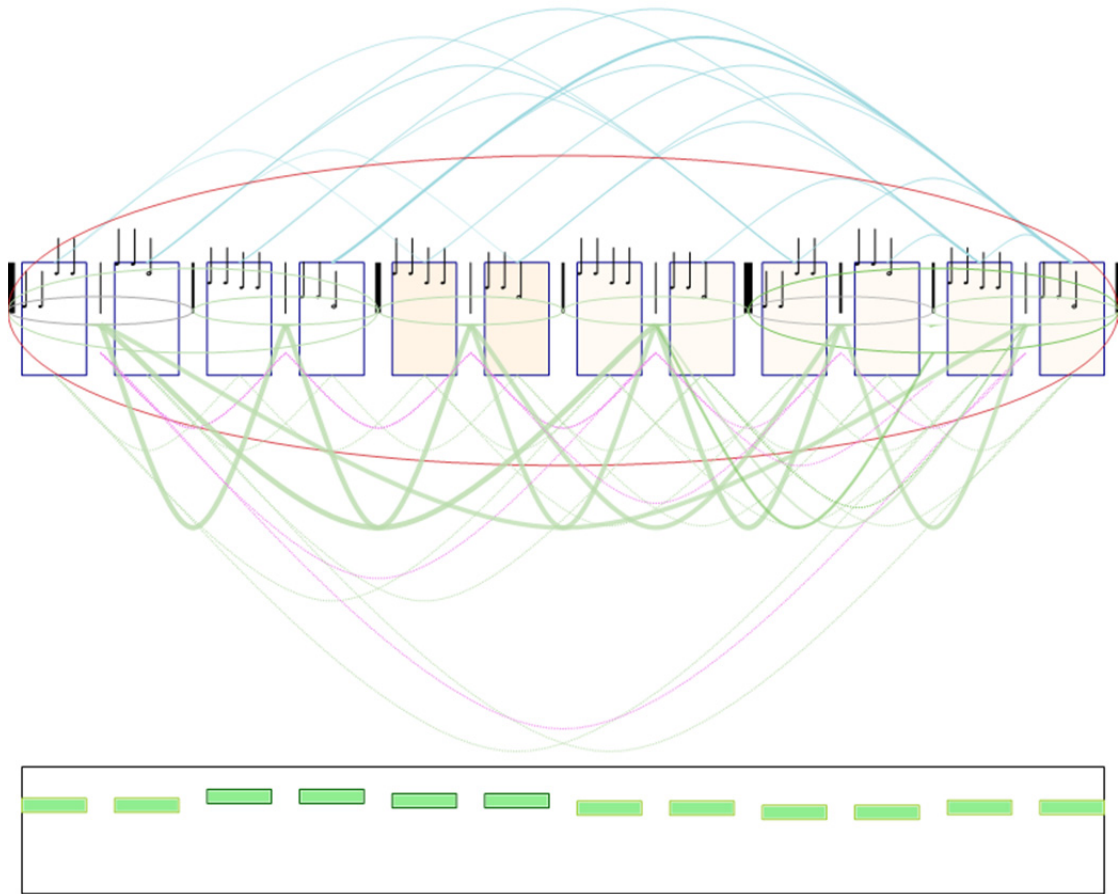


Figure 5.19: Another run on "Twinkle, Twinkle".

Bad Melodies

In the remainder of this chapter I present five “bad melodies”. For each melody I describe how it was constructed and then show Musicat listening to the melody. Because the program is stochastic and has different behavior each time it listens, I give examples from multiple runs for some melodies, to illustrate some of the possible variations in listening behavior.

BAD MELODY #1



Figure 5.20: Bad Melody #1 (random rhythms and notes).

I wrote a simple computer program to generate this melody using a pseudorandom number generator. The program always generates 32 notes. For each note, the pitch is chosen randomly from the white keys of the piano between middle C and the next C an octave higher. The duration for each pitch is selected at random from the following list: whole, half, quarter, eighth, sixteenth, dotted half, dotted quarter, or dotted eighth. (Ties are visible in the notated music as a result of notes starting in strange places.) No rests are allowed (except at the end of the final measure). The time signature is fixed to 4/4.

The melody sounds just as random and unstructured as one would expect, given the algorithm that generated it. I find the notes confusing and unmemorable — it indeed sounds like a series of random notes, undeserving of the term “melody”. So what happens when Musicat listens to it?

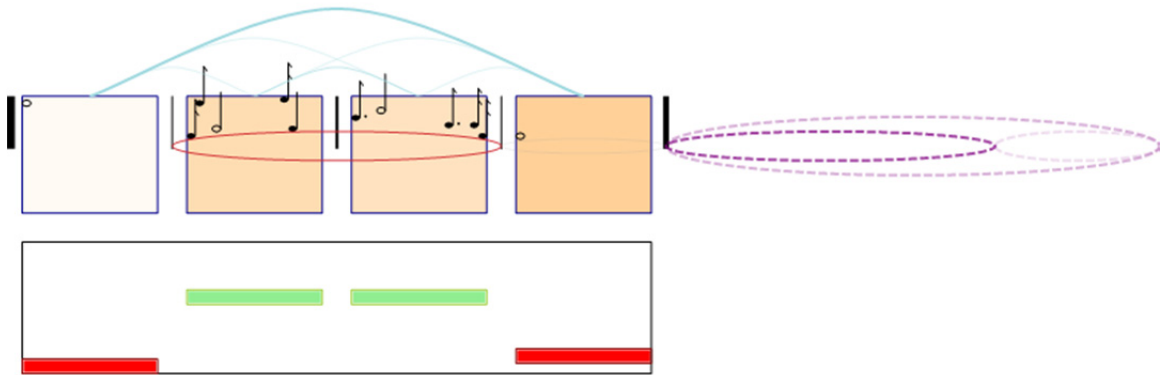


Figure 5.21: Bad Melody #1, measure 4.

The figure above shows the program's state after measure 4. This picture looks very similar to the one after measure 4 in the "Twinkle, Twinkle" run above: a group (2–3) has formed, leaving out measures 1 and 4. However, the program has noticed a link between measures 1 and 4: they are both whole notes.

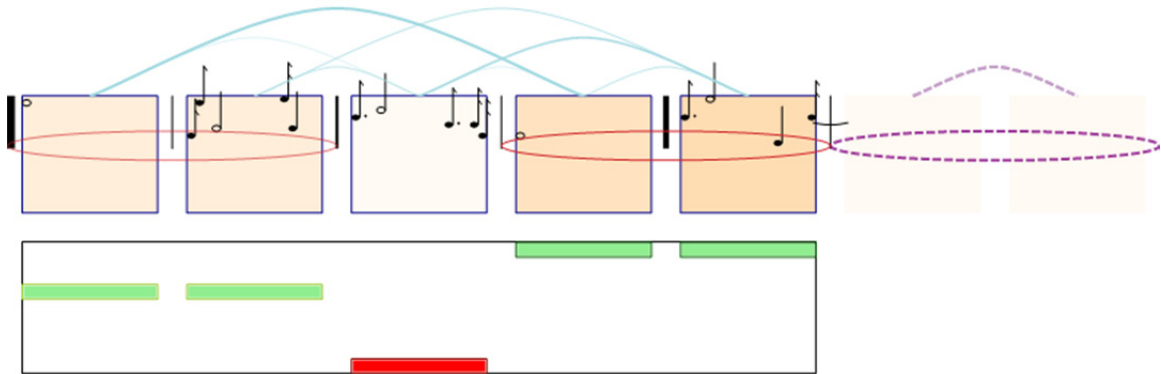


Figure 5.22: Bad Melody #1, measure 5.

One measure later, the picture has changed radically. Groups (1–2) and (4–5) are now in the picture, leaving an ungrouped measure 3 in the middle. Neither group is very strong. The strongest relationship found is the measure 1 to measure 4 connection found earlier. A new connection between measures 3 and 5 is also significant; the two measures start with the same rhythm.

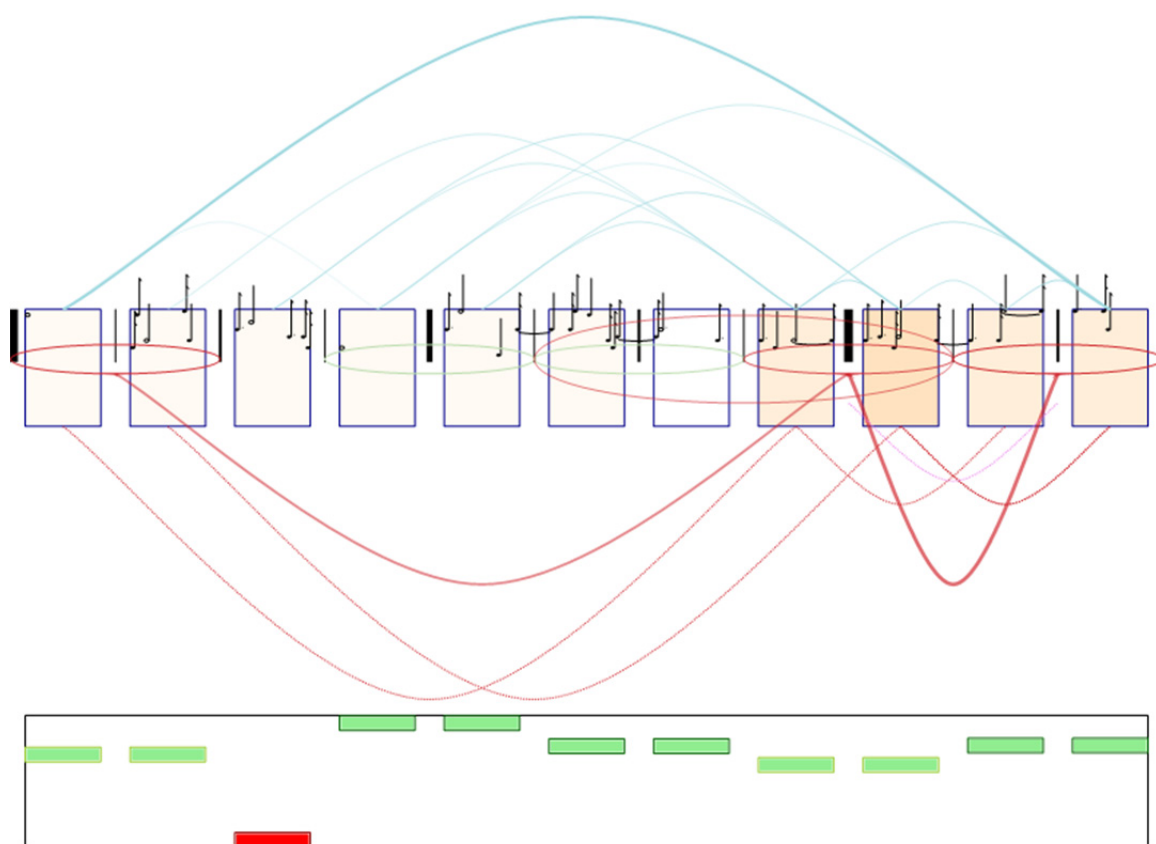


Figure 5.23: Bad Melody #1, end of processing.

Jumping ahead to the end of the listening performance, we see that the two groups present just after measure 5 was heard have survived in the program's representation. Likewise, measure 3 remains ungrouped. Several more groups have been found, as well as two analogies. It is hard to make sense of why the program has created these structures. Is this a bad listening performance? Or simply a good listening performance for a bad melody?

It turns out that the structures that have been found are quite weak. Musicat has a feature of the user interface — the “detail slider” — that allows the user to adjust the detail level up or down. By default, the detail level is set to the highest value and *all* structures are displayed. When the detail level is reduced, however, only the strongest structures are

displayed. The user can adjust the level to see which structures are really important in the program's representation.

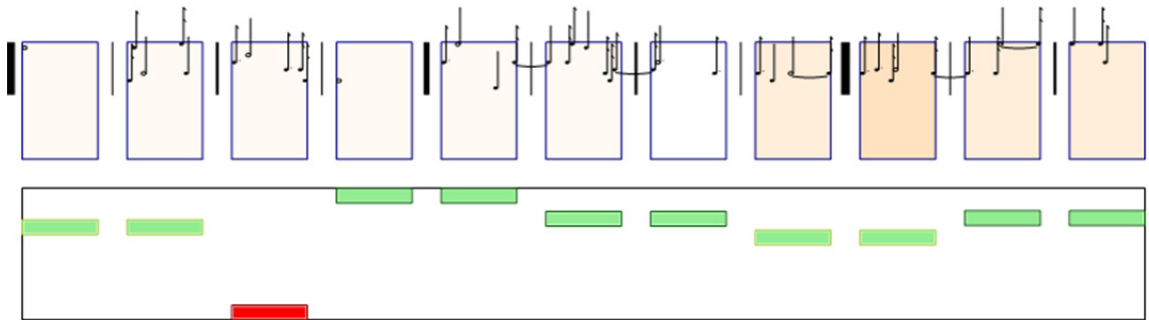


Figure 5.24: Bad Melody #1, low detail level.

When I lowered the detail level at the end of this run, every single one of the groups, links, and analogies disappeared. (Adjusting the detail level to a low-enough value will make all structures disappear for *any* run of Musicat on any melody, but in this case they all disappeared at the level that I typically use to focus attention on *strong* structures; I call this level the “low” detail level in the rest of the chapter.) Thus, *all* of the structures in this run were weak. This is as we would expect for a “bad melody”. When listening to random notes, people naturally try to make sense of the notes and attempt to pick out some structure from the randomness; Musicat seems to be doing the same thing.

In the previous figure, the heights of the happiness bars may come as a surprise: they may seem higher than expected for such a (non-)melody. This is simply an artifact of how Musicat handles working memory: groups in the distant past are made stronger so that the program avoids modifying old groups. Once the melody is complete, most of the groups are in the distant past, and thus most groups in the Workspace have a high strength value, leading to high happiness values. However, either the presence of weak analogies or a complete lack of grouping structure will reduce happiness, even for old measures (for

example, in the figure, measures 1–2 are part of a weak analogy, and measure 3 is not a member of any group, so they have low happiness values.) See Chapter 7 for more details.

Raising the detail slider to a medium setting results in the following display (the happiness bars have been hidden in this figure, as well as in analogous figures later in this chapter, because happiness values are the same as in the previous figure, and the detail slider does not affect happiness):

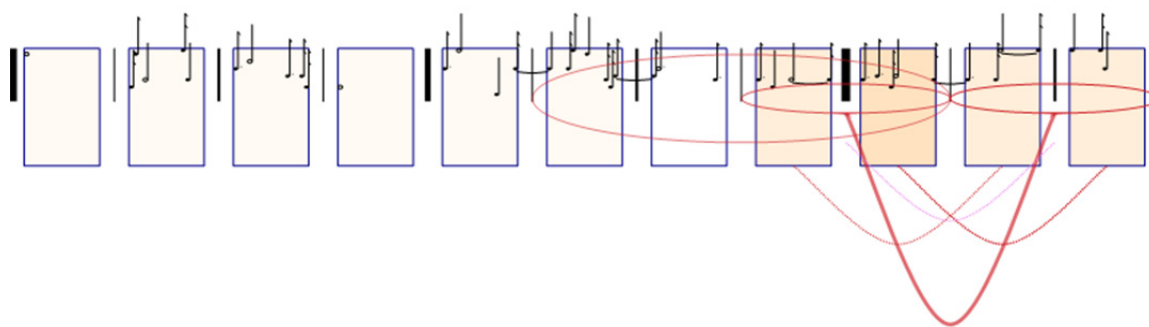


Figure 5.25: Bad Melody #1, medium level of detail.

How should one interpret this picture? These structures all disappeared under the “low” detail setting, so they aren’t extremely important to how Musicat heard the melody. Still, let’s examine what they might mean. Group (6–9) is hard to interpret. Recall that in the high-detail picture, there is a green group (6–7) inside the larger group, so this four-measure group may just be the simple result of joining the two smaller groups (6–7) and (8–9), without very many other motivating reasons for the group to exist.

The analogy (8–9) ↔ (10–11) shows three supporting reasons for its existence. The dotted pink line just below the groups involved indicates a pitch-based relationship between the two groups, although it’s not obvious from the musical score what the relationship might be. The two light-red arcs joining measure 8 to measure 10 and measure 9 to measure 11

indicate relationships Musicat discovered between these measures that support the analogy. However, it is also not obvious what these relationships are based on. Overall, Musicat's listening does not seem to tell us much about this melody — as expected, since the melody was random.

BAD MELODY #2

The previous melody was so random that it was hard for us human listeners to make any sense of it at all. Since Musicat is highly sensitive to rhythmic patterns, I decided to make a new random melody with less variation in the rhythms. Specifically, I restricted the rhythm generator by removing dotted rhythms and making it more likely for quarter notes and eighth notes to fall on non-syncopated spots in a measure. I also made the sixteenth notes show up in more regular spots: they are required to appear in tandem with another sixteenth note within the space of quarter note, in such a way that it is impossible for later rhythms to start in places that are displaced by a single sixteenth note from a strong beat (*e.g.*, a rhythm such as sixteenth-quarter-quarter-quarter, starting on a downbeat, cannot occur). Finally, I associated selection weights to each rhythm so that some patterns (such as individual quarter notes) were more likely than others to occur. Pitch generation remained the same: notes were chosen randomly from the eight notes making up a C-major scale. The result of the first run of this new melody-generation algorithm is shown in the following figure.



Figure 5.26: Bad Melody #2 (constrained rhythms).

This melody still sounds quite random to my ear, but much less so than Bad Melody #1. The melody seems to gain some internal coherence by virtue of the more regular generative process used to create the rhythm. Even though this process is random, the rhythmic vocabulary is so highly constrained that something is internally consistent about the rhythmic style. A similar statement is true for the pitch pattern of the melody; the pitches are random but are constrained to eight white piano keys. If an accidental were added to a single note of this melody, it would likely sound quite out of place.

Much to my surprise, when I played this “bad” melody for Douglas Hofstadter, he liked certain parts of it quite a lot, and even commented that it would be “a perfect theme for a Shostakovich fugue!” It reminded him in some way of Fugue 16 from Book 2 of Shostakovich’s 24 Preludes and Fugues. A few happy coincidences served to make this melody better than either of us expected. It starts on the note C and then the note C appears several times in the first two measures, helping to establish C as the tonic. (Since C appears two times in the set of possible notes {C,D,E,F,G,A,B,C’}, C does have a higher probability of selection than other pitch classes.) As the melody progresses, the random pitches give the melody a more pandiatonic⁶ character, but in the final two measures, the note A features prominently and it sounds as if the melody has modulated to A-minor.

⁶ *Pandiatonic* melodies are ones that use the *notes* of a major scale, but without regard for the tonal *functions* typically implied by the scale.

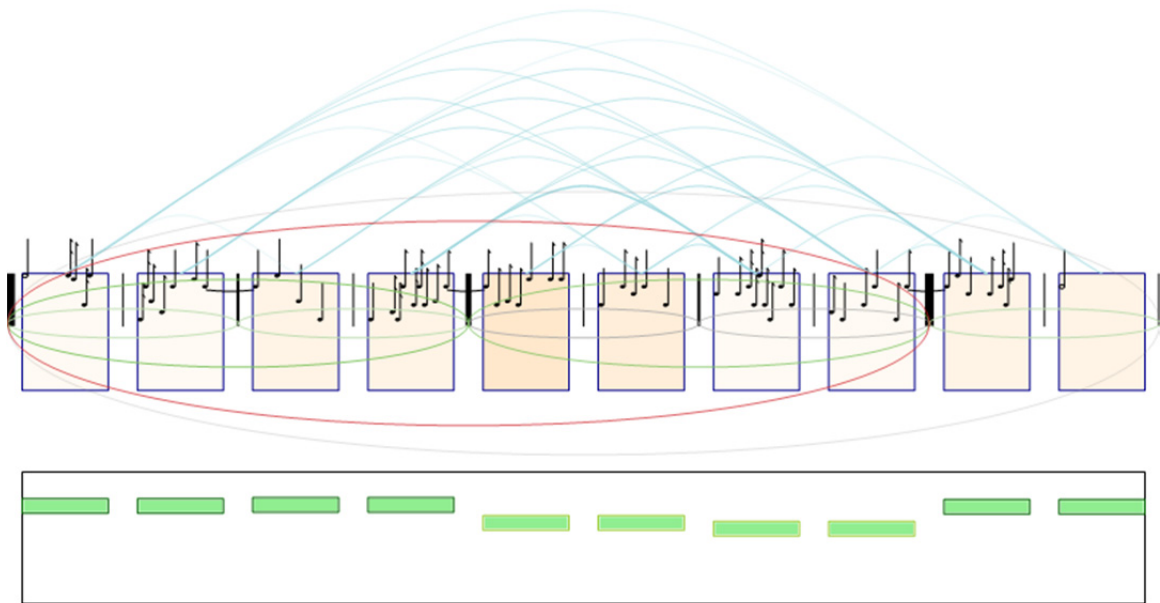


Figure 5.27: Bad Melody #2 (run 1), end of processing.

The previous figure shows the result of Musicat listening to the whole melody. (For the remainder of this chapter, I show intermediate states of the program running only when it seems particularly informative.) This listening performance has resulted in a more coherent structure than that for Bad Melody #1. Indeed, the first eight measures have been grouped in the standard binary fashion mentioned at the start of this chapter. This grouping reflects Musicat's bias to hear groups of measures in such a way that the lengths of the groups are powers of two, and pairs of adjacent groups are combined to form larger groups. In this example, the melody was 10 measures long, and we see that the final group (9–10) seems to have been tacked on somewhat arbitrarily onto the end, and included in the (very weak) meta-group (1–10) that spans all ten measures of the piece.

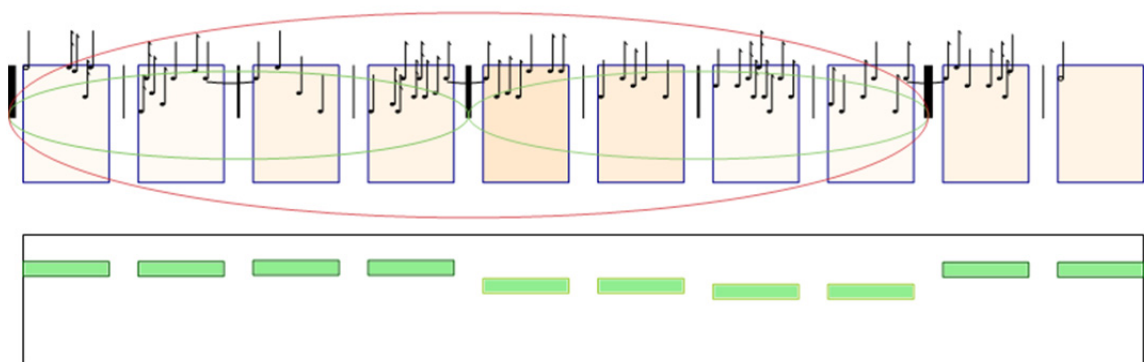


Figure 5.28: Bad Melody #2 (run 1), low level of detail.

With Bad Melody #1, turning down the detail level made all the groups disappear because they were so weak. In this case, by contrast, turning detail down to “low” leaves the two 4-measure groups and the 8-measure meta-group containing them still visible, while making all the 2-measure groups as well as the all-encompassing meta-group (1–10) disappear.

Since Musicat is nondeterministic, it was certain that different groups would form on different runs of the program. I therefore tried running it twice more; the results follow.

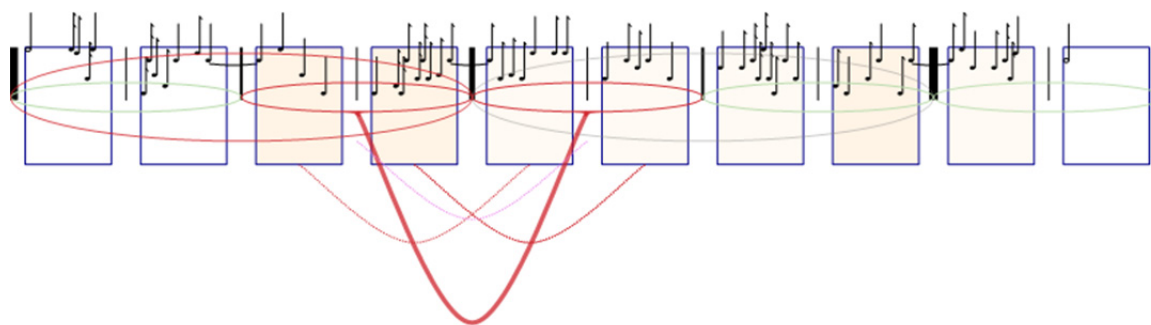


Figure 5.29: Bad Melody #2 (run 2), low level of detail.

The structures formed during this second run are similar to those in the previous run, although now five 2-measure groups are still visible even in low-detail mode. Also, an analogy

was formed this time. The analogy may be partly due to a weak melody-contour relationship found between the two groups concerned.

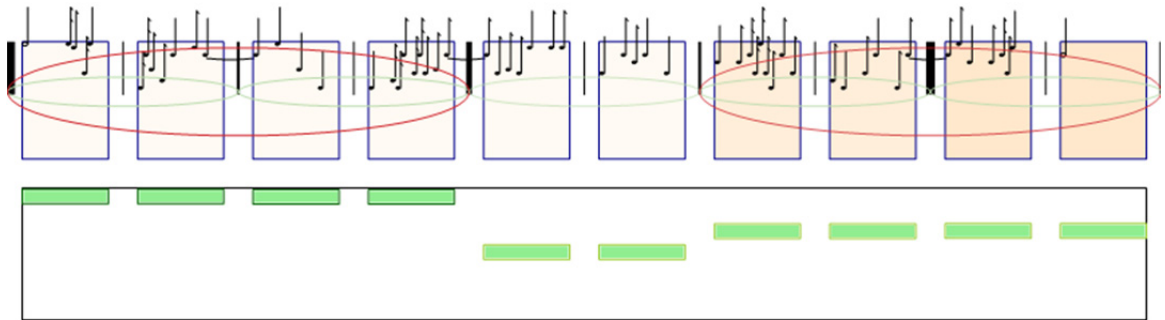


Figure 5.30: Bad Melody #2 (run 3), low level of detail.

In a third run, the grouping structure turned out a little different: the 4-measure and 2-measure groups in the final 6 measures have switched places, resulting in a symmetric grouping structure for the melody. As in the first run, no analogies were found.

Even if I didn't know about the algorithms that generated these two bad melodies, it is obvious just from observing the difference in Musicat's grouping structures that there was an important difference between the melodies. Musicat was clearly responding to some quality of this melody that was not present in Bad Melody #1. The randomness of these melodies resulted in little analogy-formation activity. But in this second bad melody, the more-structured (but still random) rhythms allowed the program's binary-structure default to become apparent in the perceived grouping structure in each of the three runs.

BAD MELODY #3



Figure 5.31: Bad Melody #3 (constrained rhythm).

I generated Bad Melody #3 using the same program that generated Bad Melody #2; the only difference was the choice of random seed. This melody makes a bit less sense to me than the previous one (although, amusingly, at the start of measure 5 the opening of the melody *Frère Jacques* appears, quite by accident— *Frère Jacques* appears on purpose in the next chapter as one of the Simple Melodies).

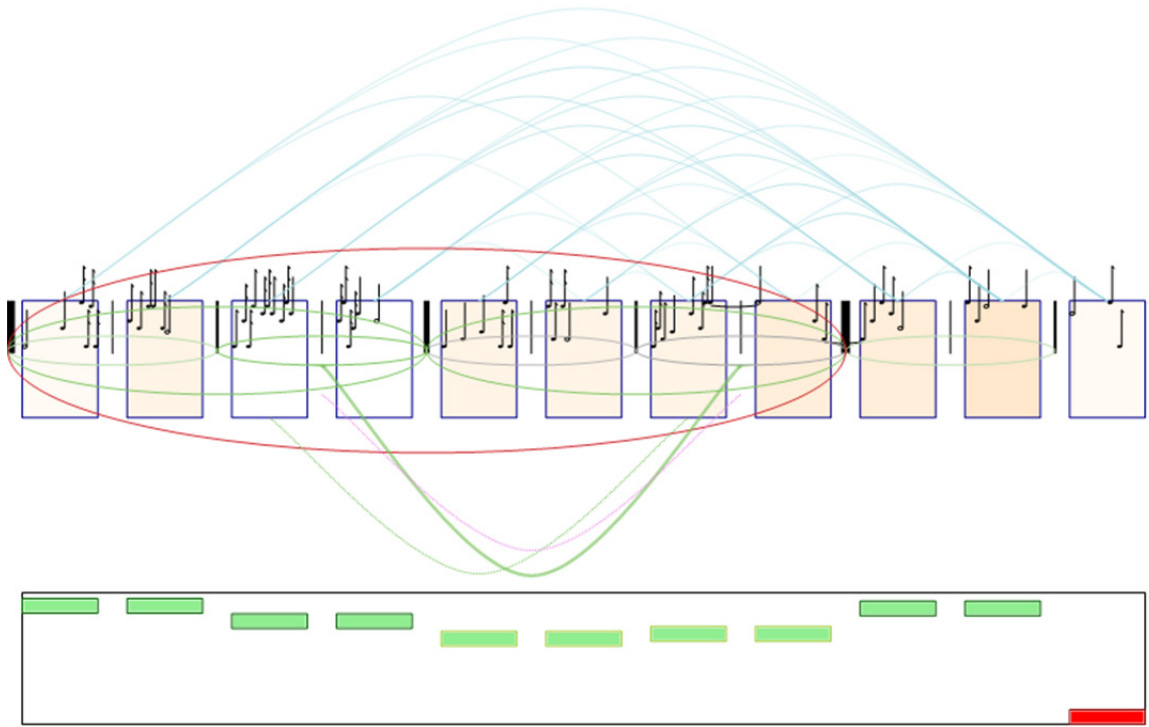


Figure 5.32: Bad Melody #3 (run 1).

Does Musicat “hear” this melody any differently than the previous one? The groups are nearly the same. This melody is one measure longer, and measure 11 has not been included in any group. An analogy has formed, $(3-4) \leftrightarrow (7-8)$, supported by a contour relationship as well as by a rhythmic relationship between measures 3 and 7. However, the analogy is not very strong; in the low-detail view it is not visible:

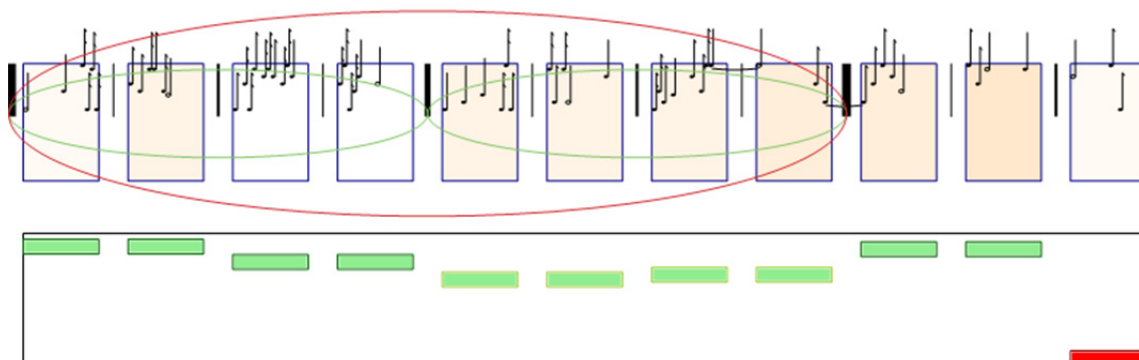


Figure 5.33: Bad Melody #3 (run 1), low detail.

This picture in the diagram above looks very similar to two of the pictures for Bad Melody #2. Analogous 4- and 8-measure groups show up here.

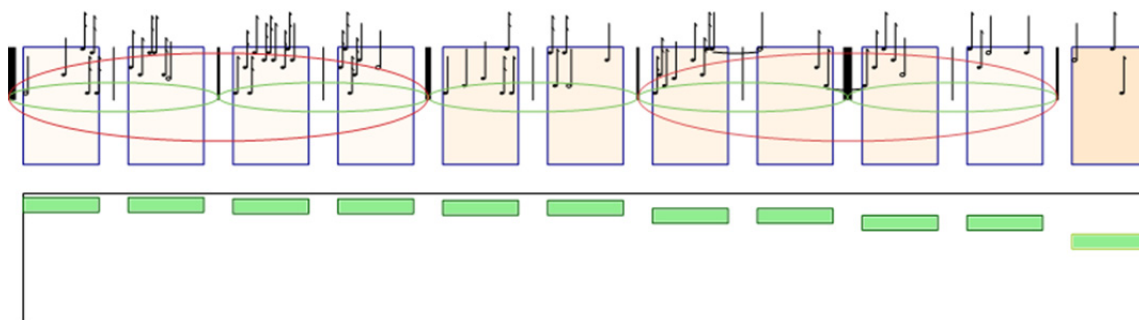


Figure 5.34: Bad Melody #3 (run 2), low detail.

Similarly, in a second run, we find the alternate symmetric grouping structure that we encountered in Bad Melody #2 (run 3), and again no analogy has formed.

Musicat attempts to find structure in these random melodies, resulting in these characteristic patterns. It is a bit surprising that it “hears” groups where we might not expect any, but it’s possible that human listeners would also hear structures such as these when given these random melodies. Less surprising, however, is that strong analogies were not formed

frequently. In the next chapter, we will see that for typical melodies, Musicat finds many more analogies, and they are often very strong.

BAD MELODY #4



Figure 5.35: Bad Melody #4 (odd-length phrases).

This melody was not composed randomly, and unlike the others in this section, it is not *bad* in the same sense of sounding very disjointed. Instead, I myself deliberately created it to have group boundaries that occur in surprising places but that should be obvious to a human listener. This is achieved by using very simple repetition to form groups, but then forcing the groups to have odd numbers of measures. Also, notice that this melody is a distortion of the *Frère Jacques* melody that appears later in this chapter. I would expect a human to hear the melody as having the following groups: (1–3), (4–6), (7–9), (10–11), and perhaps higher-level groups (1–5) and (7–11). The 2- and 3-measure groups are particularly obvious because each one consists of either exact repetition, for groups (1–3), (7–9), and (10–11), or a simple sequence (a repetition of each measure, with transposition), in the case of group (4–6). Can Musicat overcome its bias favoring groups with lengths that are powers of 2?

For this melody it is illuminating to see what happens at intermediate stages in the listening performance.

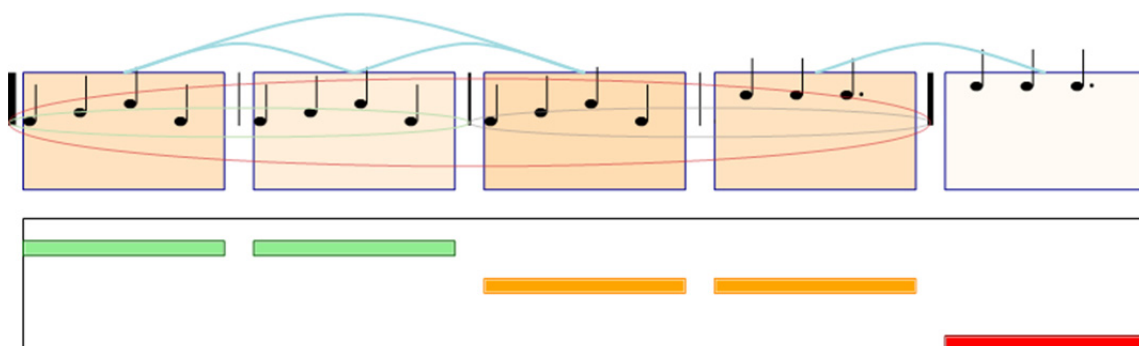


Figure 5.36: Bad Melody #4, measure 5.

After hearing measure 5, the program has created two 2-measure groups, as well as a meta-group (1–4) containing these two groups, so it seems as if the powers-of-2 bias is overpowering the natural grouping structure of this melody. However, the happiness rectangles show something interesting: whereas the first two measures are relatively happy (the happiness bars are green), the second two measures are less happy (orange bars). Group (3–4) is weak, resulting in the low happiness values. Also, the final measure has minimal happiness (red bar) because it is not included in any group.

Despite the questionable grouping structure displayed here, Musicat has noticed the obvious similarities in the measures, which are shown by the measure links above the measures. Notice that measures 1, 2, and 3 are all linked together, as are measures 4 and 5. All the clues are in place for Musicat to overcome its default pressure to form groups of lengths 2, 4, 8, 16, and so on.

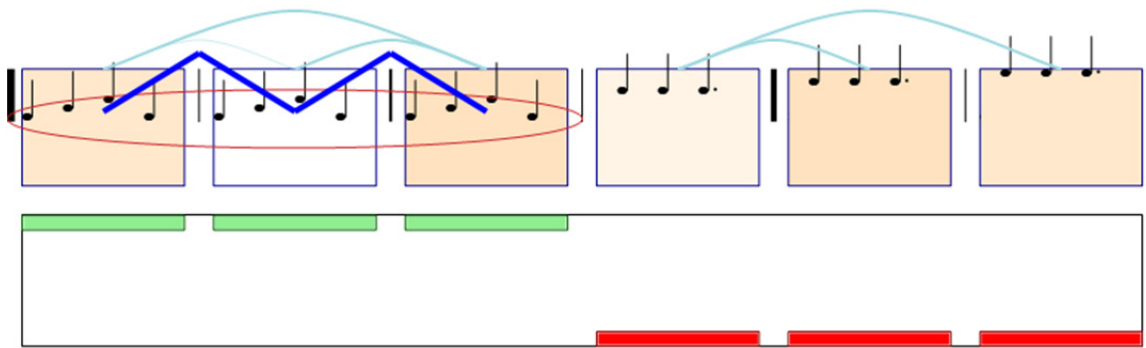


Figure 5.37: Bad Melody #4, measure 6.

After measure 6 has been heard, something radical has happened! A group of length 3 was indeed created as desired. The thick blue angled lines in the figure indicate that Musicat has created a *sequence* spanning measures 1–3. This came as a surprise to me, because Musicat’s definition of “sequence” requires successive tonal transpositions between measures; typically a sequence would involve all the notes in each successive measure shifting up or down by a scale step. However, in Musicat there are no constraints on the size of the transposition, so it “heard” these measures as successive transpositions by 0 steps!! A measure followed by two copies of itself is, to be sure, a type of sequence, albeit a trivial one.

An additional measure link has formed to link measures 1 and 3, but the next expected group (4–6) has not yet formed.

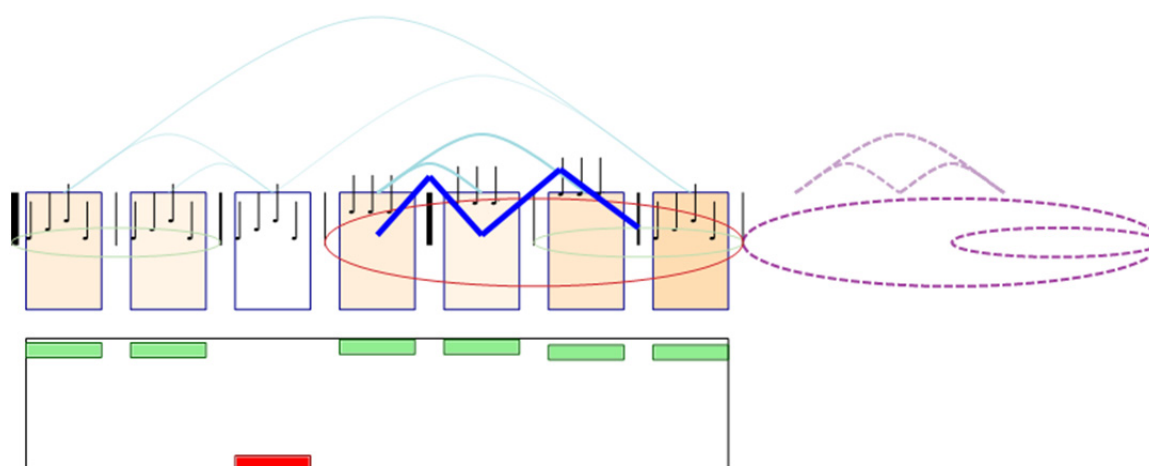


Figure 5.38: Bad Melody #4, measure 7.

After measure 7 is heard, we have both good and bad news. Another sequence has been formed, but the old sequence has disappeared, apparently having lost a competition to the initial (1–2) group, leaving measure 3 ungrouped and unhappy. The new sequence is, curiously, not made out of three measures, as would be expected. Instead, the third element of the sequence is the peculiar group (6–7), so the sequence itself constitutes a 4-measure meta-group. This is explicitly allowed by Musicat’s definition of sequence: the final element of the sequence is allowed to be a group that has extra material tacked on to the end, resulting in a group with longer duration than earlier elements of the sequence (this often happens at the end of sequences in traditional music: for example, a melodic pattern might appear in three successive measures, but the pattern in the final measure would be extended to fill out a 4-measure phrase). In this case, however, the group (6–7) is certainly not one we expected to form.

Another problem visible here is the thick bar line between measures 4 and 5. Ideally, the sequence starting with measure 4, in concert with the sequence ending right after measure 3, would cause a thick bar line to appear between measures 3 and 4, reinforcing that

spot as a place where groups should end or start. Unfortunately, though, in the figure, the thick bar line is in the “wrong” place and weakens the perceived strength of the initial sequence for Musicat, making the desired grouping structure less likely to form and become stable.

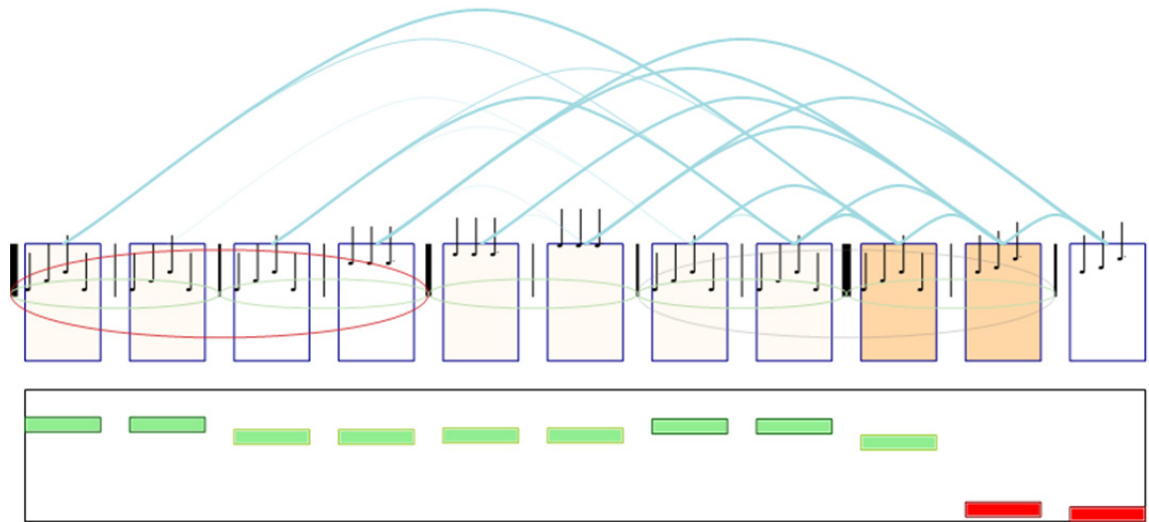


Figure 5.39: Bad Melody #4, end of processing.

After hearing all 11 measures, the program has reverted to a very stereotyped 2- and 4-measure group structure, unfortunately. The sequences did not persist through the run. I tried another run, with similar results, in which the hoped-for sequences would appear briefly, only to be destroyed by a rival length-2 group. The final grouping structure is again one of the familiar structures we saw in Bad Melodies #2 and #3. Also, no analogies were formed (although many strong measure links were discovered).

This run, though disappointing, is interesting because it gives a glimpse into the internal pressures that drive how Musicat listens. In this case, the pressure for length-2 and length-4 groups clearly overwhelmed the pressure to hear groups made up of exact repetitions

and sequences, but during the middle of the listening performance, it was not clear which pressures would prevail in the end.

BAD MELODY #5**Figure 5.40: Bad Melody #5.**

This melody starts with four rather arbitrary measures (measures 1 and 2 seem to go together, but measures 3 and 4 seem like independent musical thoughts. These are followed by several measures of other melodies, simply cut-and-pasted together. Measure 5 is from the start of “Good People All” (which appears in a later section), measures 6–7 are based on “Younger than Springtime”, and the final two measures come from the end of “Twinkle, Twinkle”.

Since this version of Musicat does not have a long-term memory that allows it to recall musical motifs from pieces it has heard in the past, it is not in the least influenced by the melodic allusions in this example. For a human listener, of course, such reminders would very likely influence the way this melody is heard, probably resulting in a sensation of amusement or even hilarity, based on the utter incongruity of the quotations’ meaningless juxtaposition.

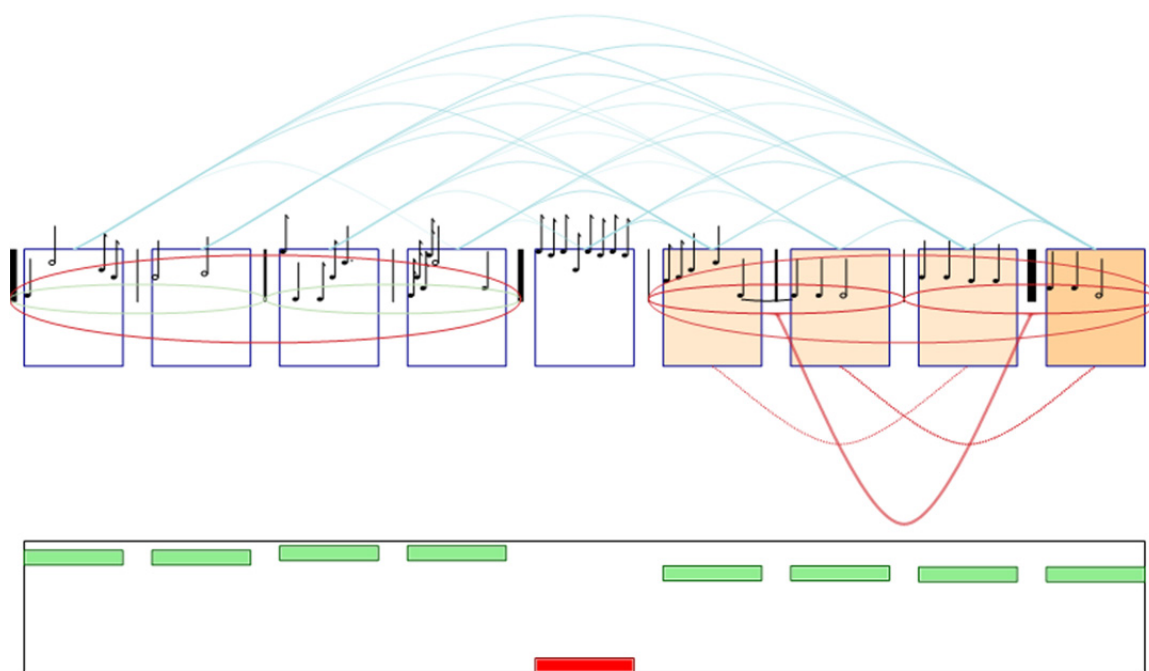


Figure 5.41: Bad Melody #5 (run 1).

The results of the first run on this melody surprised me. The program made an analogy that I didn't expect: $(6-7) \leftrightarrow (8-9)$. I didn't expect it because I had created the melody by arbitrarily pasting measures 6–7 from one melody and measures 8–9 from another. The analogy makes sense, though: the rhythm of measure 7 is very similar to that of measure 9 and they both end on the note C. The program has also found the not totally unreasonable group $(1-2)$, and group $(3-4)$ also seems comprehensible. Most interestingly, measure 5 is not part of any group, despite the program's unhappiness with this situation (which would constitute a pressure to try to find a way to include it in a group). This seems reasonable; measure 5 is saliently different from all the other measures in the melody. Perhaps a more human-like listening would include this measure as the only member of a trivial 1-measure group. Overall, though, this listening performance by Musicat seems human-like in many ways.

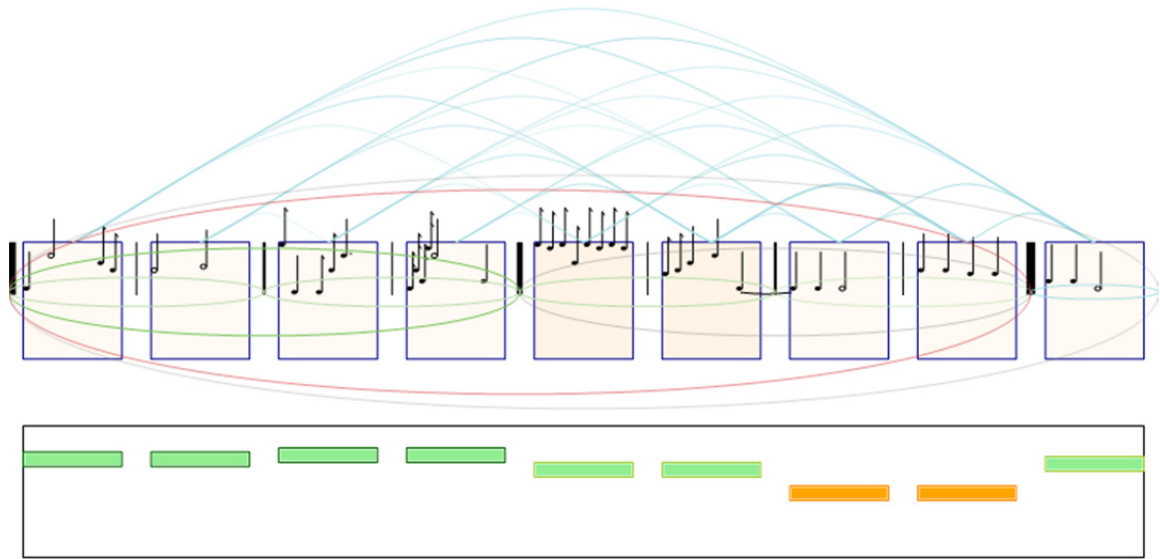


Figure 5.42: Bad Melody #5 (run 2).

On a second run on the same melody, the program found a different structure. It has simply grouped the first 8 measures in a regular binary fashion, with measure 9 tacked on at the end to make a large (but very weak) 9-measure group. Measures 7 and 8 have low happiness values, but they nonetheless form a group, even though the group (8–9) would have been much stronger. This run is less satisfying to me than the previous one was, because it doesn't seem to parallel a human listening experience nearly as well as that run did.

SUMMARY

Musicat gave mixed results on these five “bad melodies”. In places its listening performance seems similar to a human's. Specifically, for Bad Melody #1 and Bad Melody #5 (run 1), it behaved essentially as I hoped. In Bad Melody #4, the program made some of the expected structures during the run, but disappointingly, they did not persist all the way to the end of the run. In the other examples, the program exhibited a preference for hearing melodies in terms of a regular binary structure, but it is highly questionable whether such structure was warranted for these random melodies.

Even though Musicat made some *groups* that humans might not have created, in most of these examples, the program did not find many strong *analogies* between groups. Such a lack of analogies is to be expected for random melodies. In the next chapter, by contrast, we will see how Musicat typically finds *many* analogies in simple melodies; its listening performances for random melodies will (fortunately) turn out to be very different from its listening performances for more typical, coherent melodies.

CHAPTER SIX

Musicat Listens to Simple Melodies

This chapter demonstrates Musicat listening to the most basic type of melodies we had in mind when we first envisioned the program. These are rather short melodies with structures that are easy to understand for a human listener; many of these melodies are children's songs. Of course, music listening is hard, as was discussed in Chapter 1, and even simple melodies like these can exhibit complex and elusive structures. In short, even the very simple melodies discussed here constitute enormous challenges.

A key feature of Musicat is that, like other programs using the FARG architecture, it follows different pathways and comes up with different results on different runs. Thus, it is important to see how the program behaves on multiple runs. I showed multiple runs on some of the Bad Melodies in the previous chapter, but in this chapter it will be even more important to show how Musicat behaves on different runs. It would not be sufficient for me to simply show the results from a single run for each melody, because you, the reader, would not be assured that I had not simply picked the "best" run from a dozen or a hundred different runs. Moreover, there is not, in general, a single "best" way of listening, just as in the Copycat domain there is no single "correct" answer to a given analogy problem. Musicat notices different musical features and forms different groups and analogies on different runs.

We hope, however, that obvious and strong relationships will stand out and will be formed by the program on nearly every run. Therefore, for each melody in this chapter, I have presented results from at least two different runs, and I took care to show *typical* results, which give a realistic picture of how the program typically behaves, rather than outlier performances that might give an exaggerated sense of Musicat’s “talent” at listening to and understanding music.

TWINKLE, TWINKLE, LITTLE STAR (RUN 2)

This section shows the results from another run of the program on “Twinkle, Twinkle”, our sample melody from the start of the previous chapter. However, in this run I will skip the measure-by-measure displays, and will just show screenshots of the program at a few selected points during its listening.

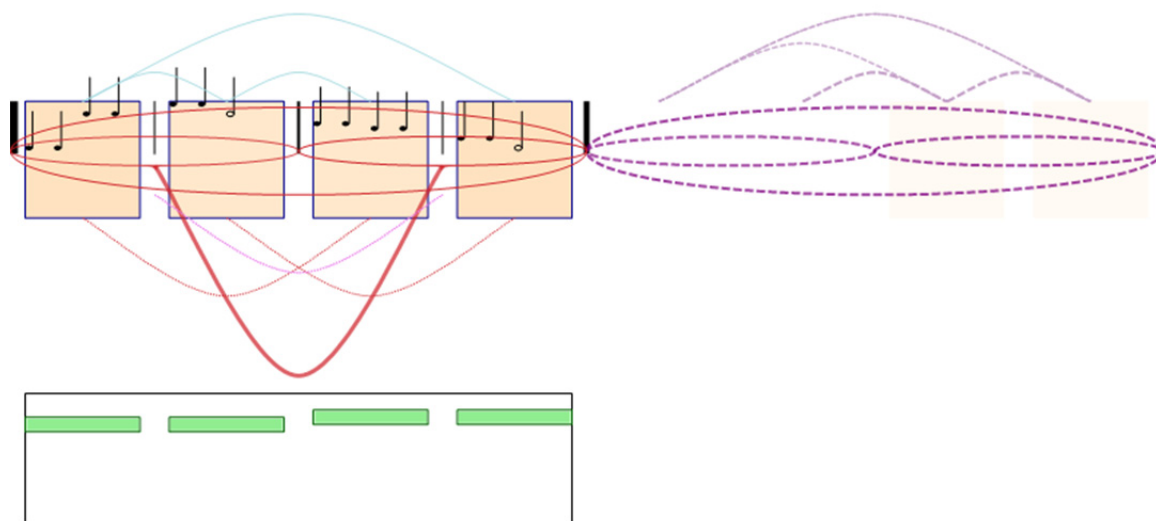


Figure 6.1: Twinkle Twinkle (run 2), measure 4.

After measure 4 has been heard, the expected groups have formed, as has an analogy between them. In the previous run, the program was confused at this point; in this run, on the other hand, everything is proceeding as expected.

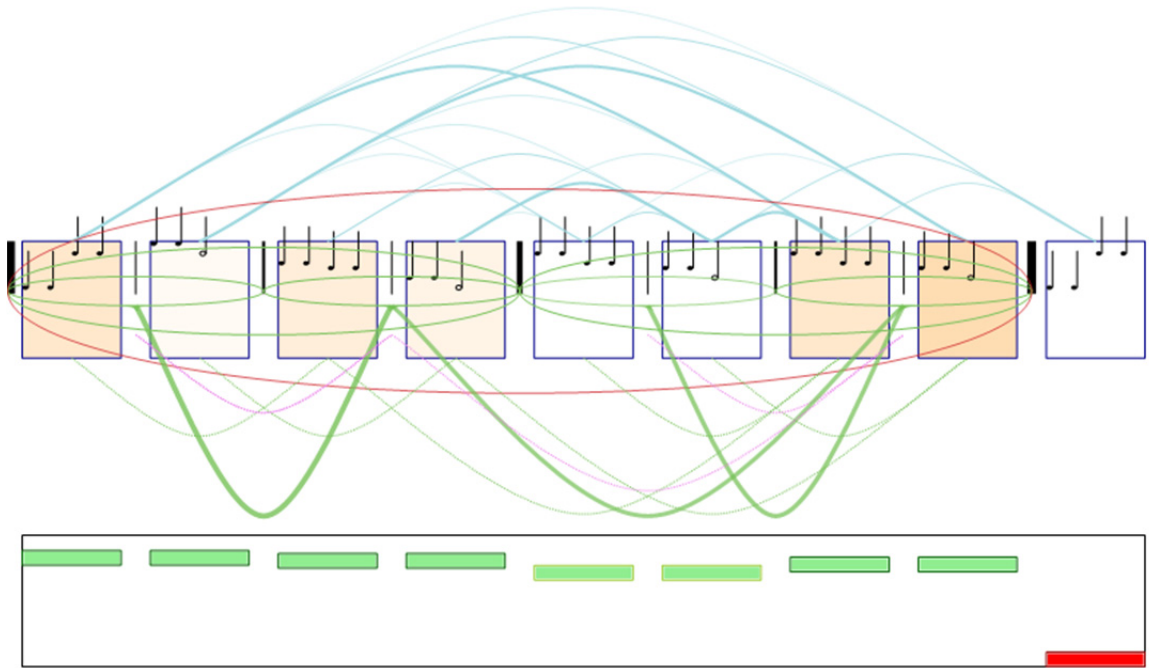


Figure 6.2: Twinkle, Twinkle (run 2), measure 9.

After measure 9, many analogies have been formed (notice how this strongly contrasts with the results for the “bad melodies” in the previous section). In the first run of “Twinkle, Twinkle”, at this point in the listening performance, the analogies discovered were $(1-2) \leftrightarrow (7-8)$ and $(1-2) \leftrightarrow (5-6)$. Both involved the initial group $(1-2)$. In this run, by contrast, there is only one analogy involving the initial group: $(1-2) \leftrightarrow (3-4)$. But notice how similar *all* of the two-measure groups are after the first two measures: $(3-4)$, $(5-6)$, and $(7-8)$ look virtually identical. Indeed, $(5-6)$ and $(7-8)$ are exactly the same, each of them being a simple transposition of $(3-4)$ up one step. Thus it is not so surprising that in the earlier run, Musicat made an analogy from $(1-2)$ to $(7-8)$, while on this run $(1-2)$ was

mapped to (3–4) instead. — one could imagine swapping most of these measure-pairs around in the melody, and regardless of the order, there are obvious relationships between all pairs. Measure (1–2) is the most unique in this group, so it requires a bit more flexibility in analogy-making to map it onto any of the other groups.

In this run, Musicat finally saw the strongest and most obvious analogy amongst this set of 2-measure groups, which was missing in the previous run: it made the analogy between the two identical measure-pairs (5–6) and (7–8). It also noticed the analogy based on the transposition mentioned above: (3–4)↔(7–8). But why, one might then wonder, didn't it also make the analogy (3–4)↔(5–6)? Indeed, it might find this analogy some of the time. However, the program slightly prefers making a correspondence between the two groups ending on measures 4 and 8, respectively, because they both end just before a thick bar line. In other words, it notices how these two groups are situated similarly in terms of the metric hierarchy of the piece; each of them forms the end of a larger structure. Group (5–6), in contrast, is at the start of a large structure, and so there is some justification for Musicat's making the (3–4)↔(7–8) analogy more often than the (3–4)↔(5–6) analogy.

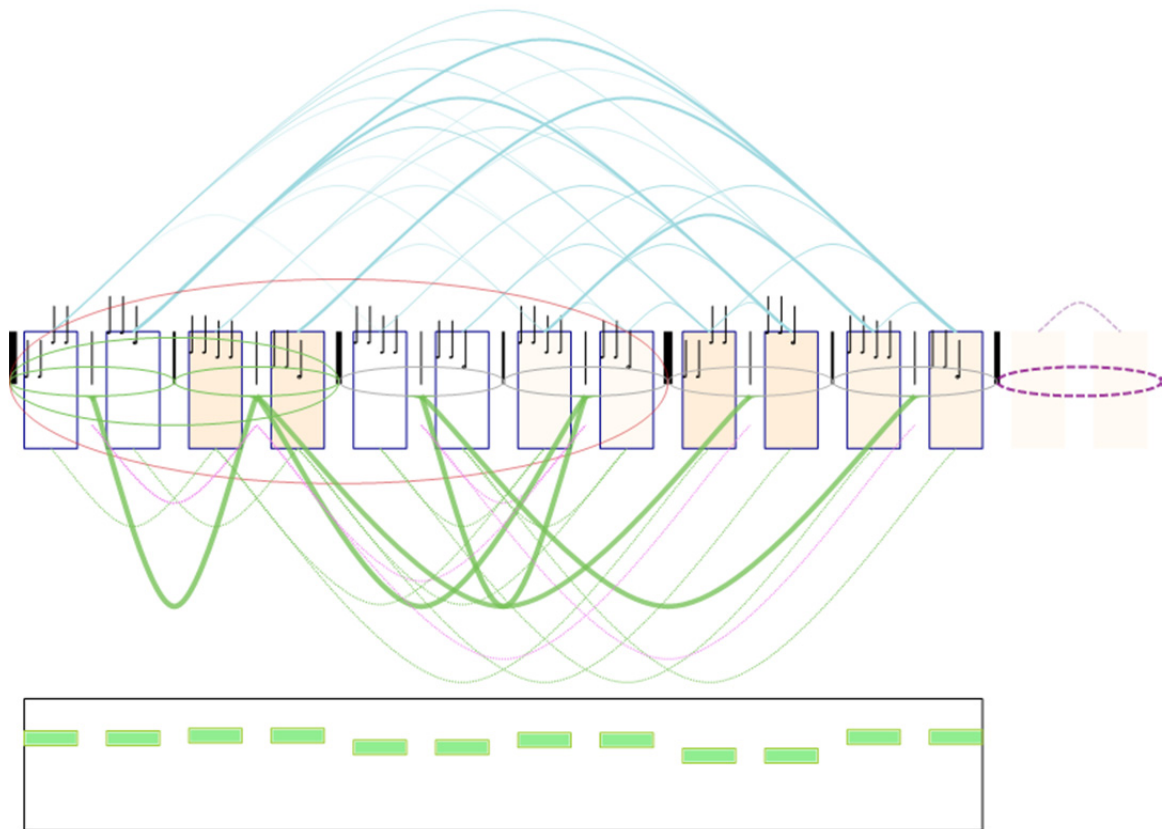


Figure 6.3: Twinkle, Twinkle (run 2), measure 12.

In the figure above, all 12 measures have been presented to the program. In addition to the analogies in place after measure 9 was heard, two new analogies have formed, linking the new groups in the final four measures to earlier groups. The grouping structure has also changed slightly. Group (5–8) has disappeared (or is extremely weak and essentially invisible). The large eight-measure group (1–8) is still in place (despite the missing or weak inner group (5–8)), but the expected final group (9–12) is missing (indeed, since the final four measures are identical to the first four, we might well expect Musicat to create the same grouping structure). However, this listening performance is not yet complete: the program always goes on to “think” about the music it has just heard for the time of several additional

measures. And in this particular run, it turns out that a lot will happen during this final processing time.

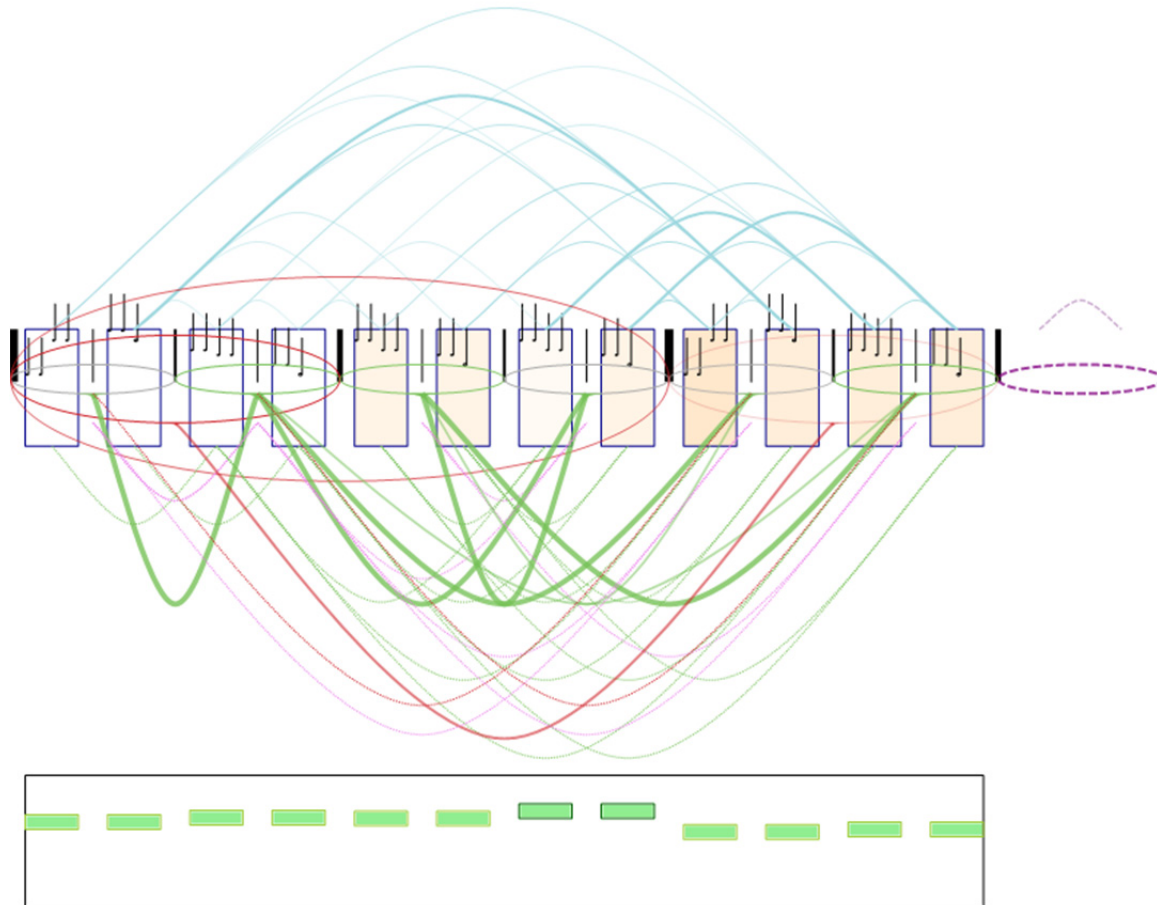


Figure 6.4: Twinkle, Twinkle (run 2), measure 12 (+3 measures of extra time).

After three more measures of time have passed, group (9–12) has formed, most gratifyingly, and even more gratifyingly, a new long-distance analogy (colored red) links this new group to the group comprised of the first four measures: (1–4) ↔ (9–12). This picture is quite similar to the one at the end of the previous run, although in this run there are more

small analogies (colored green). Also, the lack of group (5–8) is conspicuous; that central group was involved in two analogies in the first run.

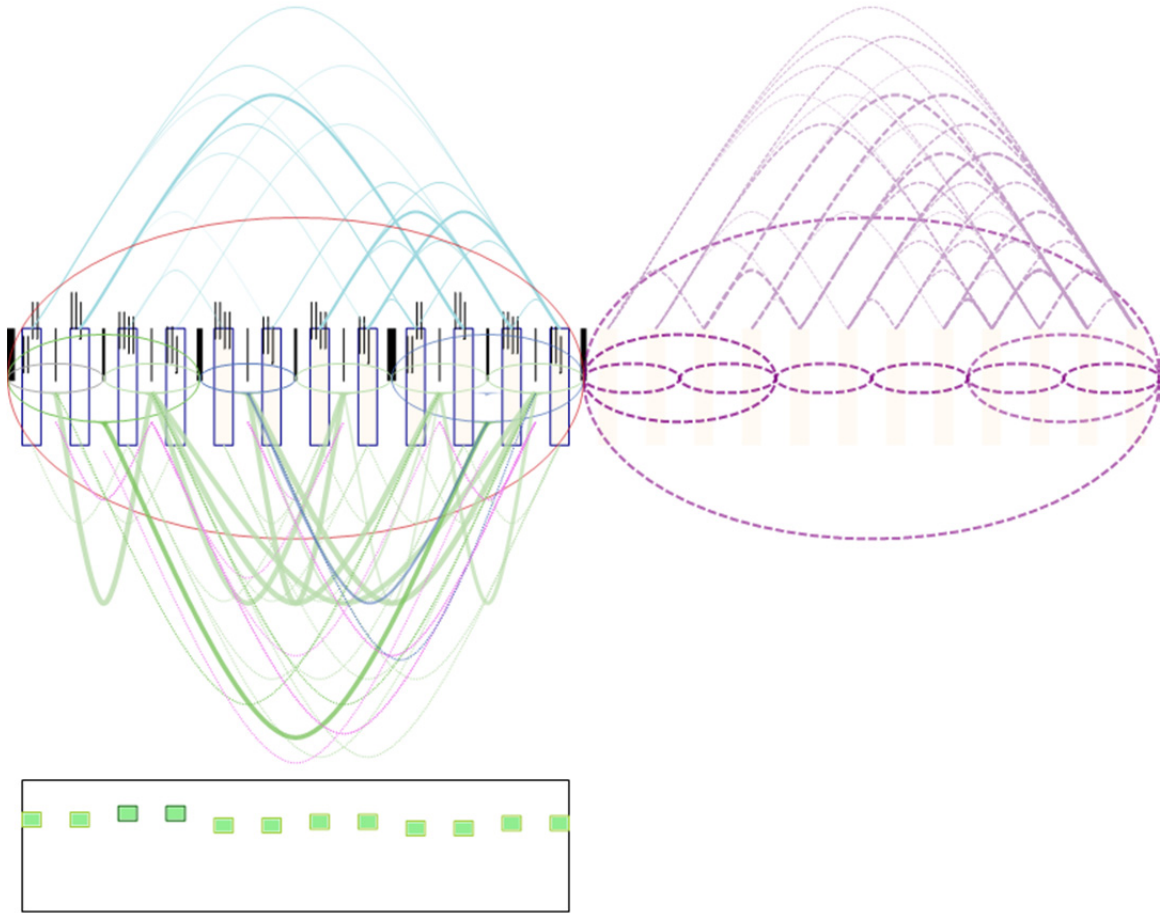


Figure 6.5: Twinkle, Twinkle (run 2), measure 12(+4).

After one more measure of time has passed, something new has happened: the entire melody has been enclosed in one large group, (1–12). Recall that this didn't happen on the previous run. Moreover, Musicat now expects the entire melody to repeat as a whole: the large purple structure at the right indicates that the program expects this same structure to occur in the next 12 bars. Because the melody has ended, however, the program will soon destroy all expectations automatically.

Many green analogies are visible, although the strongest is the relatively new analogy (1–4)↔(9–12), linking the first four measures to their exact repetition at the end of the melody. This analogy was visible in Figure 4 but it has gained considerably in strength since then, thanks to the extra processing time after the melody’s final note “sounded”.

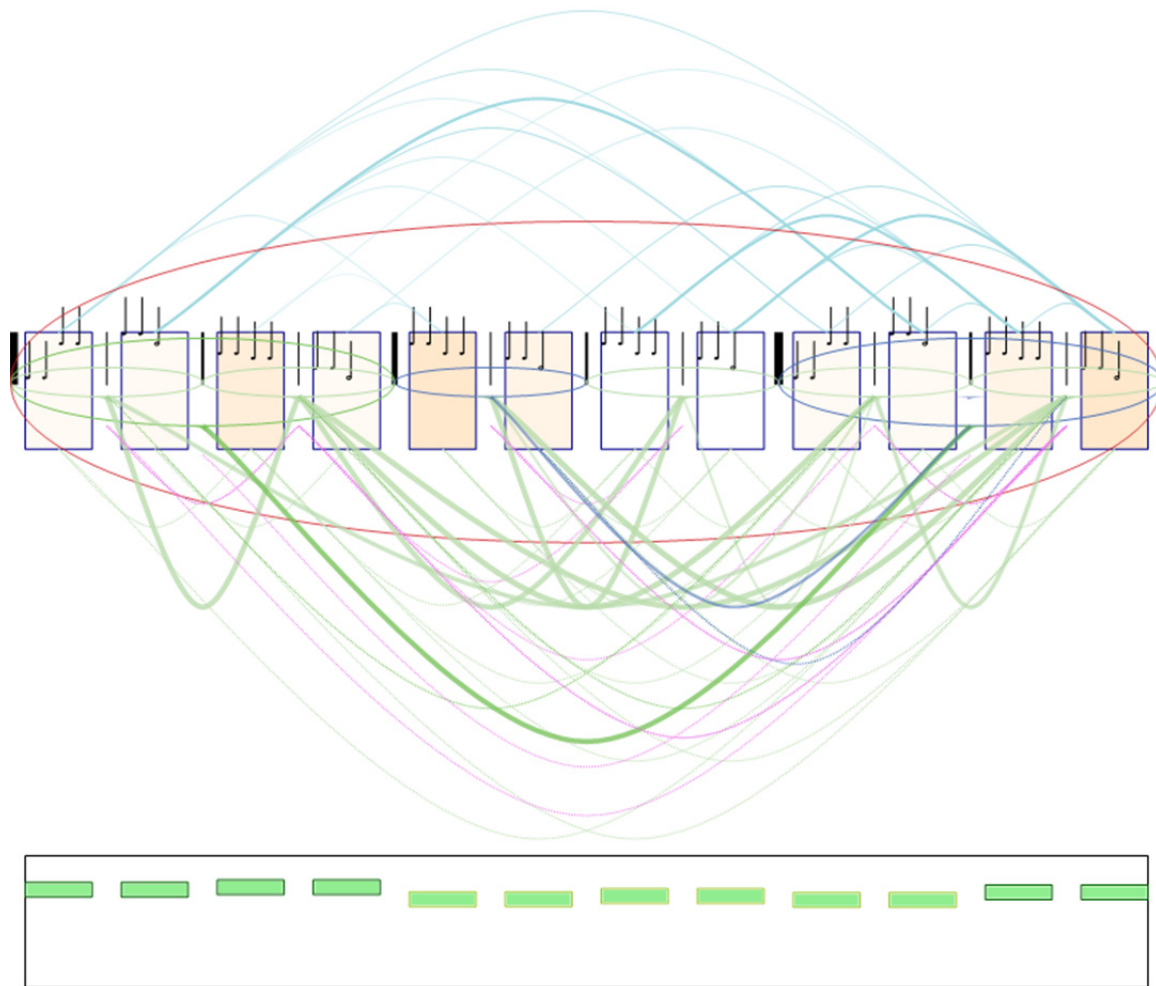


Figure 6.6: Twinkle, Twinkle (run 2), end of run.

At last, the listening is complete; the figure above shows the discovered structures at the end of the run. Because there are so many analogies, it is a bit hard to make sense of the

picture. Therefore, I will show how this picture varies when I adjust the program's detail level slider, just as I did in the Bad Melodies section. However, in this section the goal is not to identify *weak* structures as much as it is to help make the *strongest* structures more visible.

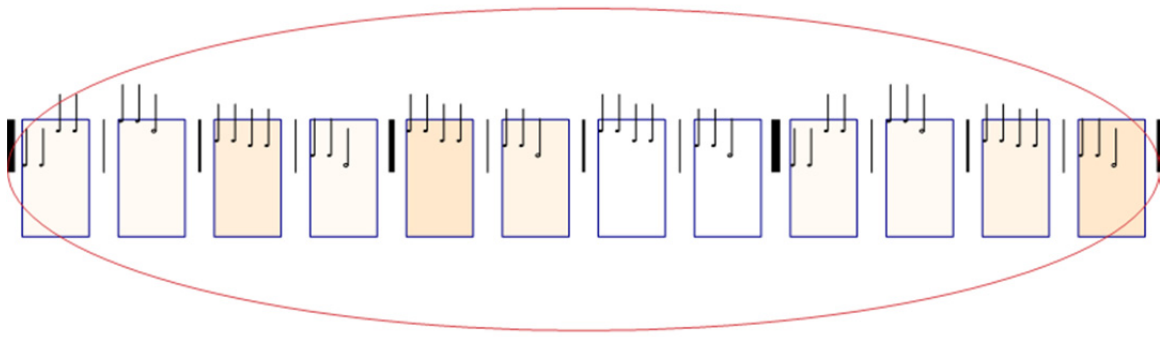


Figure 6.7: Twinkle, Twinkle (run 2), outer group (*very* low level of detail).

When I turn down the detail level to a very low setting, the only remaining structure (aside from the measures, bar lines, and notes, which can never disappear) is the group encompassing the entire melody.

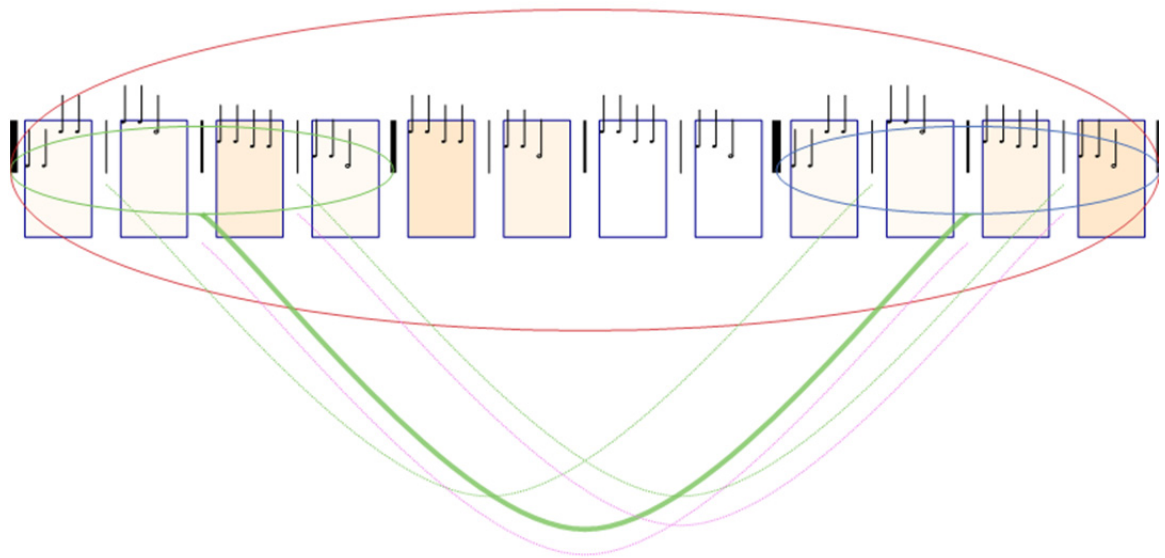


Figure 6.8: Twinkle, Twinkle (run 2), big analogy (low level of detail).

Turning up the detail level slightly results in the strong long-distance analogy appearing. This picture helps us understand what Musicat has heard: fundamentally, it has heard the melody as a large 12-measure structure, with two strong internal groups, (1–4) and (9–12), which are connected by a strong analogy.

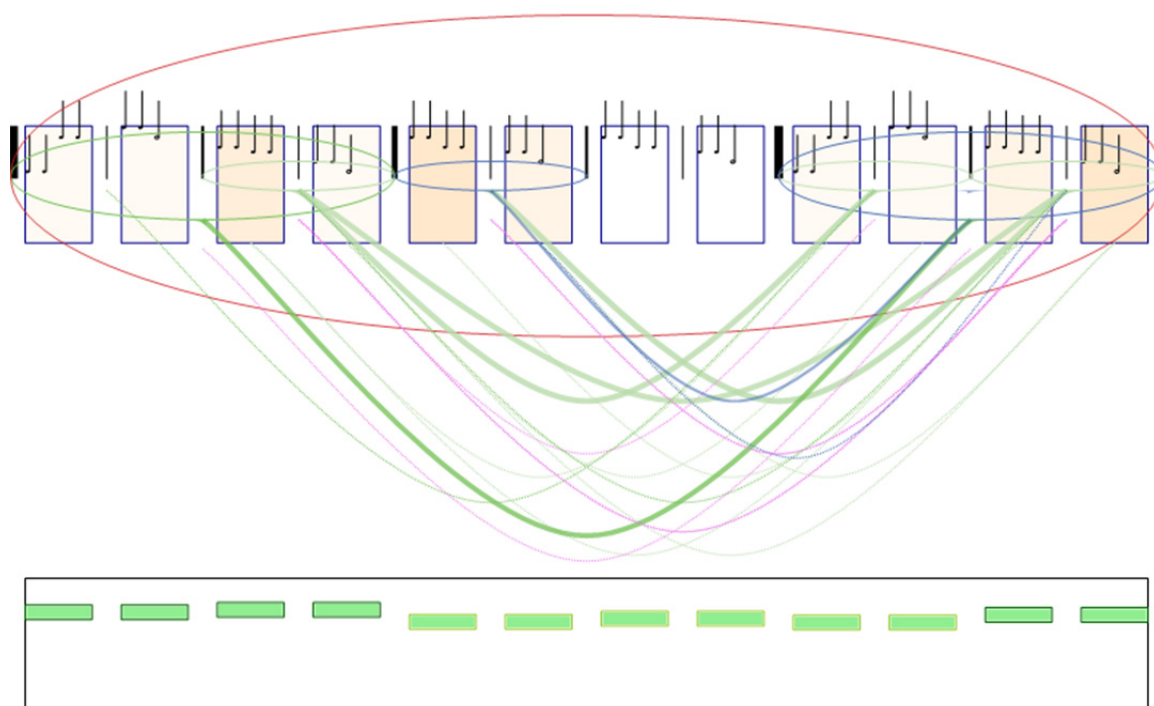


Figure 6.9: Twinkle, Twinkle (run 2), medium detail level.

Turning up the detail level a bit more results in this figure. The strongest analogies can be seen here; some of these are components of the large analogy from the previous figure. Interestingly, one of these sub-analogies involves the two-measure group (3–4), which has been mapped onto *both* (9–10) and (11–12), while the other important sub-analogy, (1–2)↔(9–10), is too weak to be visible in this view.

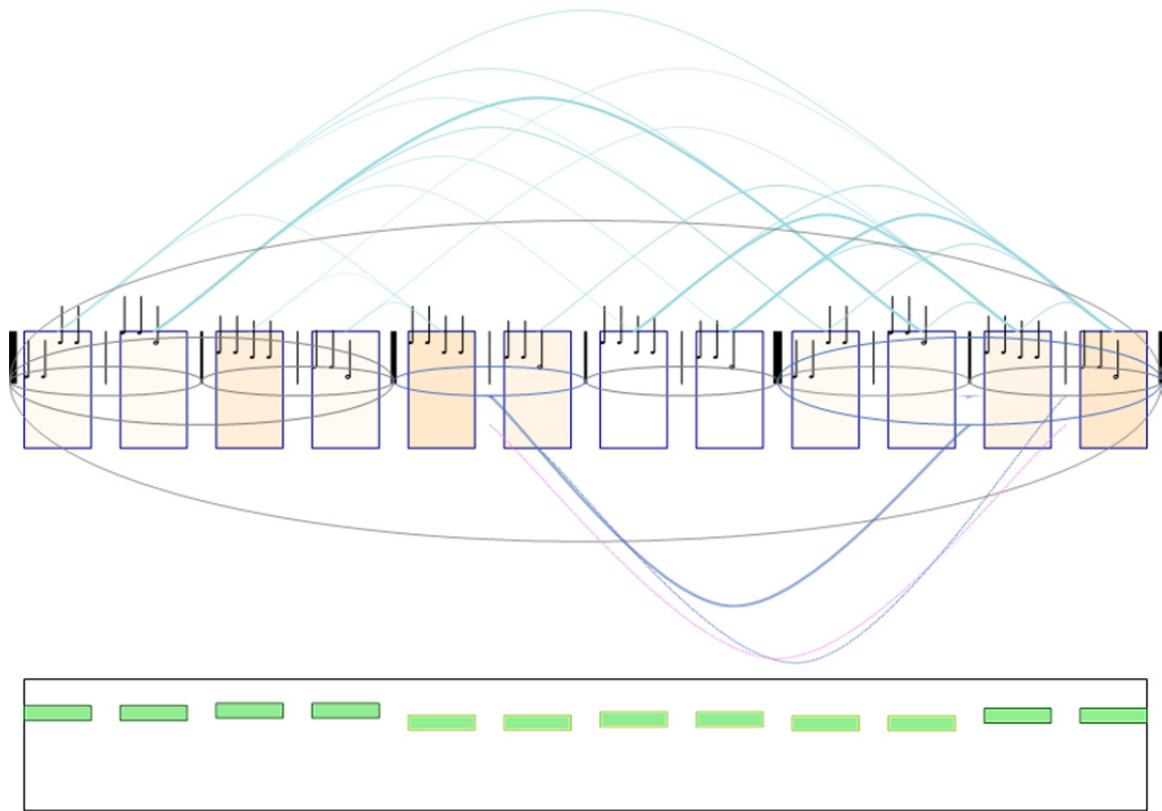


Figure 6.10: Twinkle, Twinkle (run 2), medium detail level. Analogy in blue singled-out.

In this figure, I left the detail level at “medium” but have hidden all the green analogies, in order to highlight a curious analogy, $(5-6) \leftrightarrow (9-12)$, which was made between two different-sized structures. Even though the groups involved are of different lengths, the program recognized a strong similarity. This is not too surprising; as I mentioned earlier, $(5-6)$ is a transposition of the melody in $(3-4)$, or, equivalently, its copy at the end of the melody $(11-12)$.) The figure shows (with a light blue line) that the fundamental reason for this analogy is indeed the mapping $(5-6) \leftrightarrow (11-12)$.

ROW, ROW, ROW YOUR BOAT

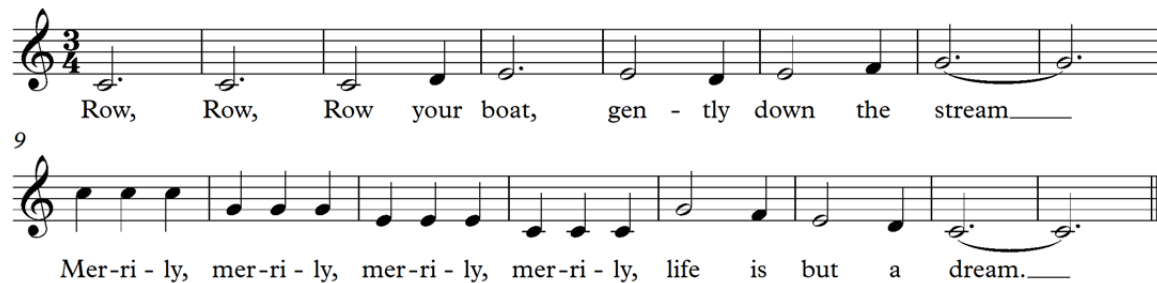


Figure 6.11: Row, Row, Row Your Boat.

The “Row, Row, Row Your Boat” melody presented a surprisingly significant challenge to Musicat: when we first gave this melody to an earlier version of the program, the results seemed meaningless. In this section I will present the results of five runs (using the current version of the program) to illustrate various things that the program notices, and fails to notice, when listening to the melody. I encourage the reader to stop for a moment, to hum or sing this melody, and to reflect on what groupings and analogies come to mind, before proceeding.

A comment on the figures in the rest of this chapter: from this point forward, all figures will use the highest detail setting unless otherwise noted. Also, because of space constraints in the user interface (recall that these pictures are simply screen captures of the program as it runs), the top of the display is often cut off for long melodies. For example, we cannot see the very top of the blue measure-link arcs in many of the figures below, but since these are the least significant structures generated by the program, that lack should not affect our understanding at all.

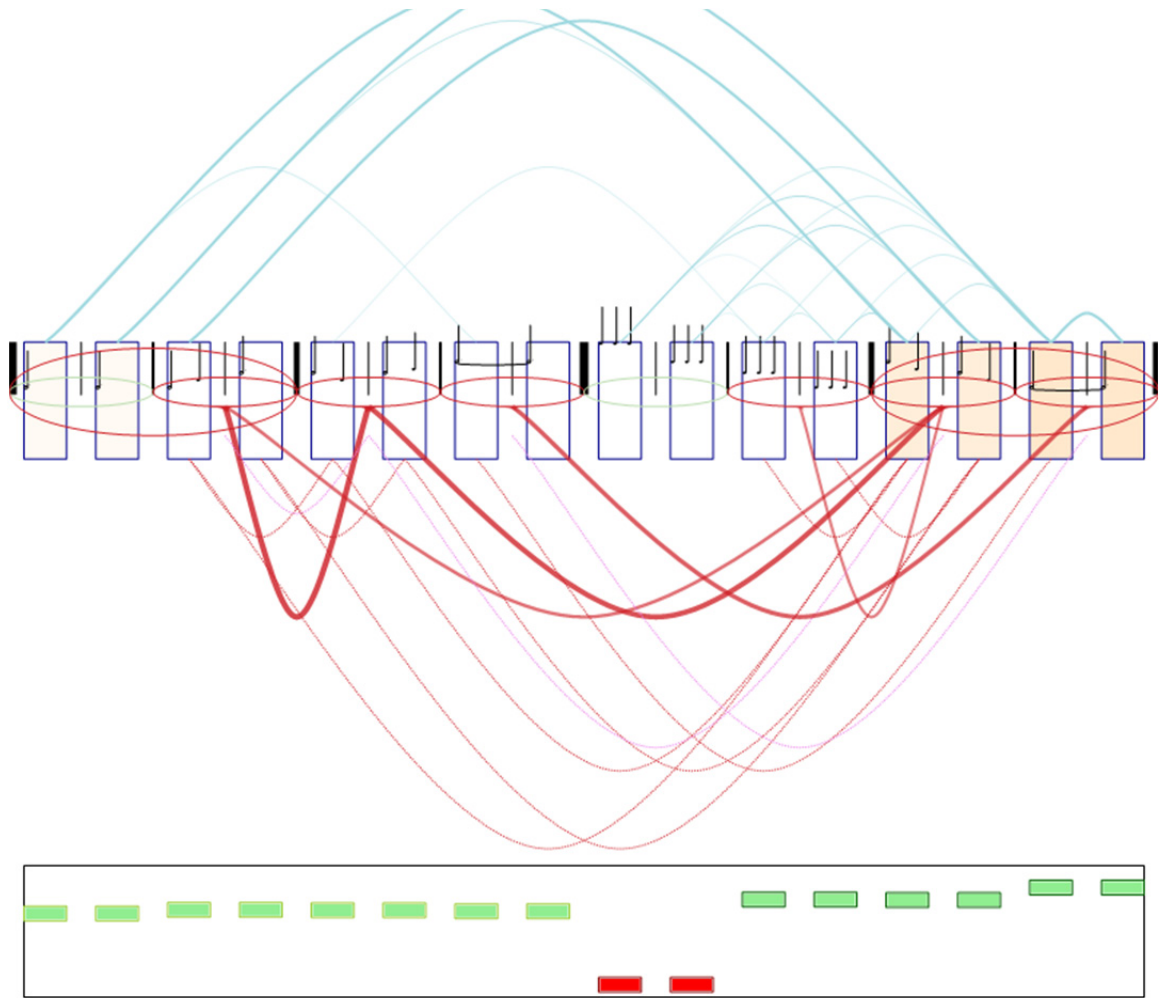


Figure 6.12: Row, Row, Row Your Boat (run 1).

This figure shows the state of the program after listening was complete for the first run. In this picture, we see that a great deal of structure has been perceived. Every successive pair of measures has been chunked into a group. The beginning and the end of the melody also exhibit 4-measure meta-groups. All the groups seem reasonable, although I expected to see more higher-order groups (every successive group of four measures sounds like a group to me).

The first and strongest analogy in the figure, (3–4)↔(5–6), is interesting and insightful. The program has noticed a contour relationship between the two groups

(indicated by the small pink arc with endpoints that are aligned horizontally with the centers of the groups involved). It is easy to see this relationship: the melody in (3–4) is moving up by steps (C–D–E), which is similar to the upwards motion of (5–6) if we ignore the first note in (E–D–E–G). The analogy is also based on the similar rhythmic pattern of the two groups: (5–6) starts with the same rhythm as (3–4) and then is extended by the addition of one more note (the quarter note at the end of measure 6). To put it in terms of the lyrics, the rhythm for the words “row your boat” is essentially reused with the following words, “gently down”. But whereas the word “boat” was set to a dotted half note, the word “down” is set to a regular half note, making room for the word “the” on a quarter note. The reuse (with slight modification) of this rhythm gives structure to the melody and propels it forward. I will have more to say about this analogy in run 4 below.

One of the most salient features of this melody to me is that I hear the notes of “gently down the stream” (5–8) as rising-upwards to the climax on “stream”, echoed by the later descent of “life is but a dream” (13–16). That is, I hear the analogy (5–8)↔(13–16). In this run, the individual two-measure groups are members of analogies, just as I would expect: (5–6)↔(13–14) and (7–8)↔(15–16). However, there is no four-measure group (5–8), just two smaller groups, and thus no single analogy (5–8)↔(13–16). In other words, Musicat *almost* formed the main structure that I myself hear and that I had hoped it would find, but it fell slightly short.

There was one more analogy made in this run: (11–12)↔(13–14), but this one is harder to explain. I had expected measures 9–12 to be involved in a different way, because their rhythm (all quarter notes) is so distinct from other parts of this melody. The very low happiness values for measures 9–10 indicate that the program was aware that it had missed

something in that section, but that problem was not fixed in this run. Let's examine a second run to see what the program does differently.

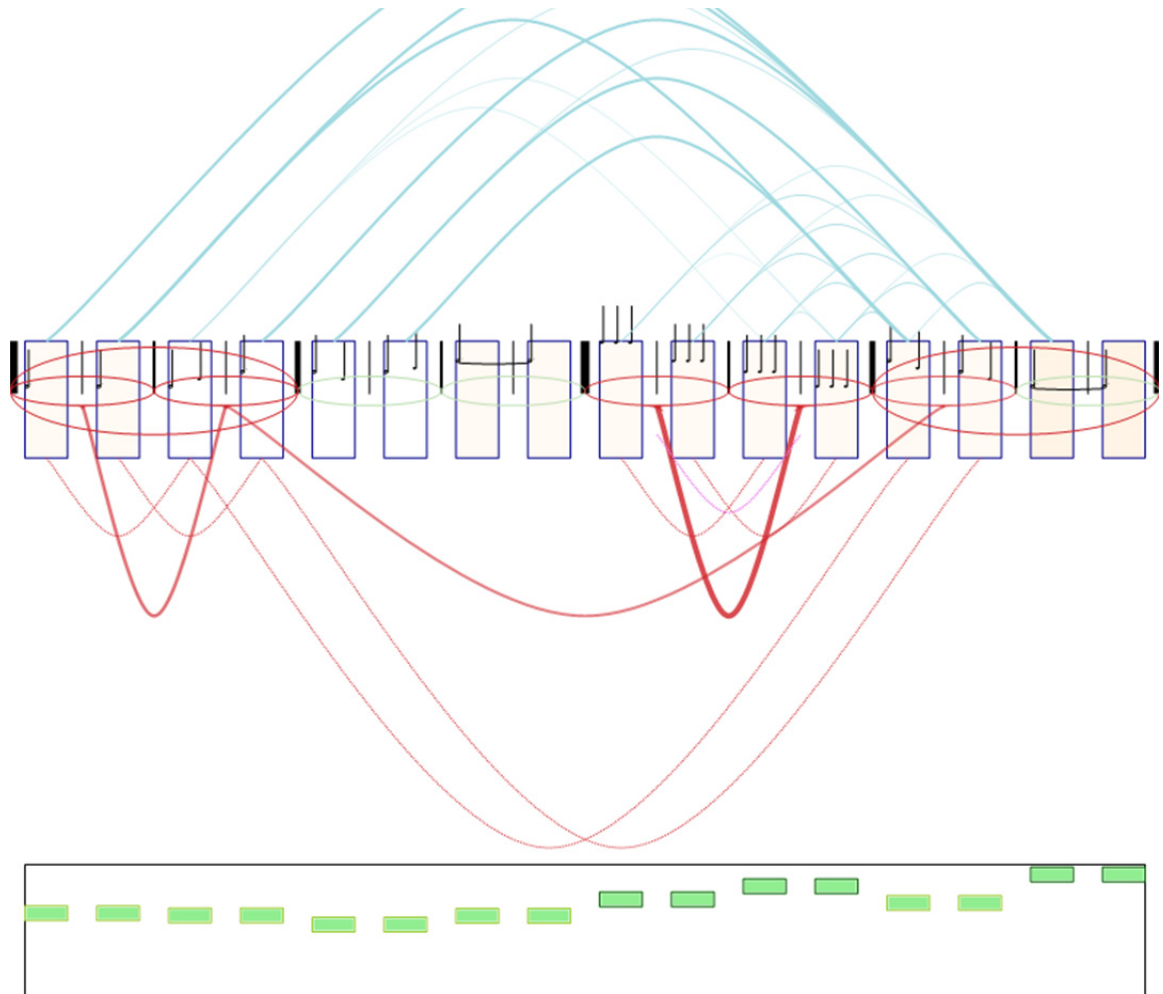


Figure 6.13: Row, Row, Row Your Boat (run 2).

In this run, the same groups were formed as in the previous run, but, curiously, the analogies are completely different. Ironically, the strongest analogy, $(9-10) \leftrightarrow (11-12)$, involves the measures that were ignored in the first run. This is the part of the melody with the lyrics “merrily, merrily, merrily, merrily”, so not only are all four measures identical in

terms of rhythm and pitch contour (each measure has a flat contour), but also, if we know the words, we hear the exact repetition of the word “merrily”. Now, I think I hear this section as four measures with three notes each, where the melody is descending through a C-major arpeggio, and so my hearing differs from Musicat’s hearing of two measures that are mapped onto the following two measures. However, the analogy found by the program is a justifiable way of hearing the passage, especially if one thinks hypermetrically, with measures 9 and 11 being more heavily stressed, and measures 10 and 12 being softer.

The program found two more analogies. Analogy (1–2)↔(3–4) is based on a rhythmic similarity between the two groups: each measure starts with a long note on the downbeat, and in measures 1, 2, and 4, this is the only note in the measure (a dotted half note). Only in measure 3 is the rhythm slightly different, with the first note of the measure shortened and a quarter note added on beat 3. Also, measures 1 and 3 both start with the note C, strengthening the analogy.

The final analogy maps two groups at quite a long distance: (3–4)↔(13–14). This is slightly different from the analogy found in the previous run, (5–6)↔(13–14). But (3–4) and (5–6) are quite similar, so it is not surprising that the program can discover both of these analogies. Indeed, both analogies make intuitive sense. (Analogies are not mutually exclusive in Musicat, but for some reason it didn’t find both of these in the same run.) Ignoring the program’s results for a moment and considering my own listening, I can easily imagine hearing an analogy between the first four and the last four measures of the melody. In this analogy, the first two measures would map onto the long, tied note at the very end of the melody, while the inner measures would be related to each other: the ascending line (“...row your boat”) in measures 3–4 is echoed at the end of the melody by the descending line (“life is but a...” in measures 13–14. Incidentally, this imagined analogy is reminiscent of simple

letter-string analogy we might make in the Copycat domain: in the mapping $\mathbf{abc} \leftrightarrow \mathbf{cba}$, the letter **a** at the *start* of the sequence is mapped onto the **a** at the *end* of the sequence, and the *ascending* pattern $\mathbf{a} \rightarrow \mathbf{b} \rightarrow \mathbf{c}$ is mapped onto the *descending* sequence $\mathbf{c} \rightarrow \mathbf{b} \rightarrow \mathbf{a}$. In the current run of “Row, Row, Row Your Boat”, Musicat did not create the entire analogy $(1-4) \leftrightarrow (13-16)$ that I expected, but it did make the analogy $(3-4) \leftrightarrow (13-14)$, which is half of my desired analogy — it maps the initial ascending line to the final descending line.

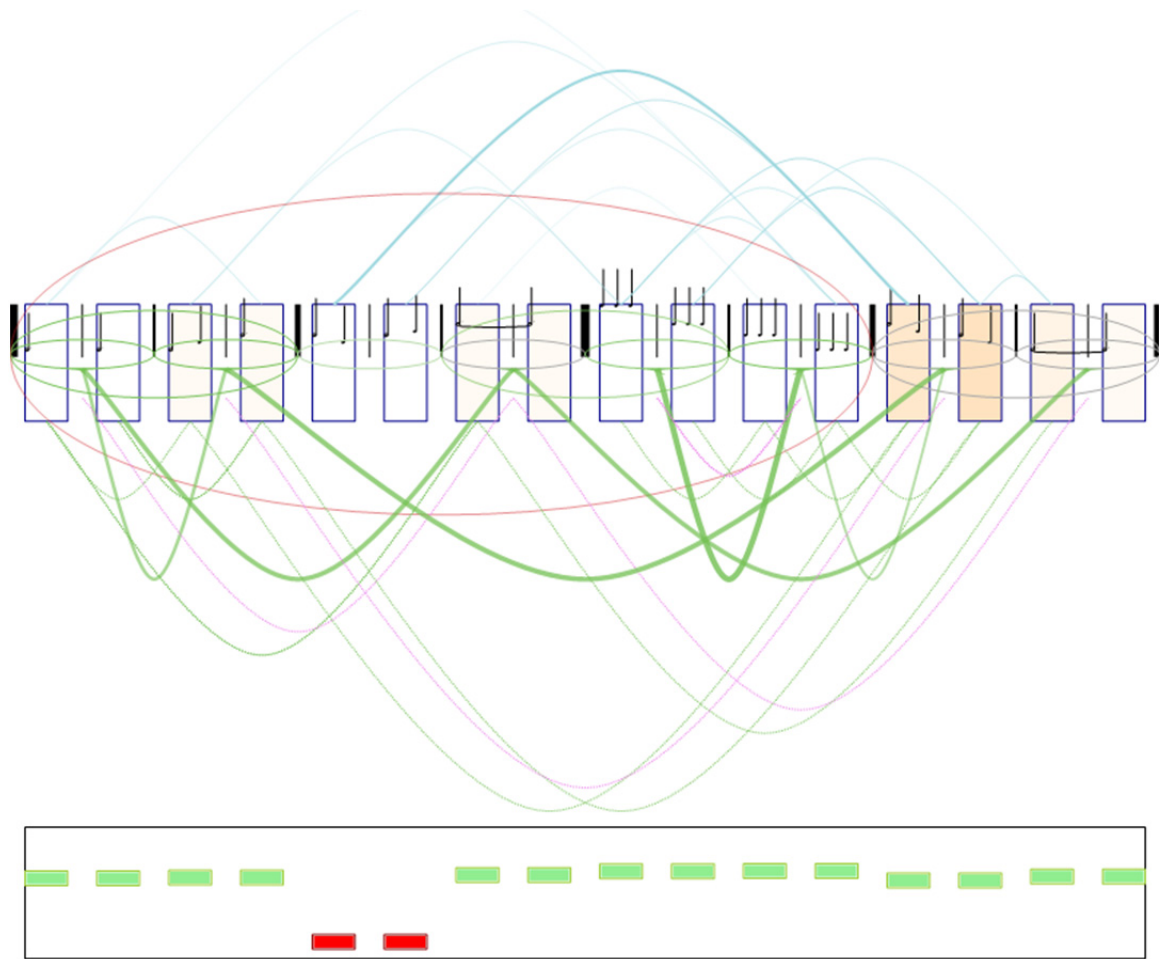


Figure 6.14: Row, Row, Row Your Boat (run 3).

In a third run, many of the analogies created in the first two runs were created again. In addition, all of the groups present before were created in this run, along with two

additional groups: the very strange meta-group (7–10), which awkwardly stretches across the thick bar line at the very center of the melody, and the large, strangely-sized group (1–12).

A new analogy in this run was (1–2) \leftrightarrow (7–8), mapping the first notes of the melody onto the long note at the end of the first half of the melody. This analogy is followed by the analogy (7–8) \leftrightarrow (15–16), which was created in the first run as well. In this run, it makes a nice, cohesive picture to see both of these analogies together, because the analogies form a sort of sequence: **A** \leftrightarrow **B** followed by **B** \leftrightarrow **C** (the program, however, doesn't have the ability to hear a relationship such as this one between two analogies). This pair of analogies helps to emphasize the large-scale structure of the melody: the opening notes are developed in such a way that they lead up to the cadence in measures 7–8, these first 8 measures form an antecedent phrase, and then the next 8 measures are a consequent phrase that ends on a cadence that is related to that which formed the end of the antecedent phrase. (The melody also has a stereotypical arch form: it rises in pitch to the middle of the melody and then falls back down to a stable base at the end of the melody. This idea about the arch shape is not noticed by Musicat, although the relationship **A** \leftrightarrow **B** \leftrightarrow **C** contains the germ of this idea, because **A** and **C** consist of notes on the tonic (the note C), whereas the **B** material is on the dominant (the note G).

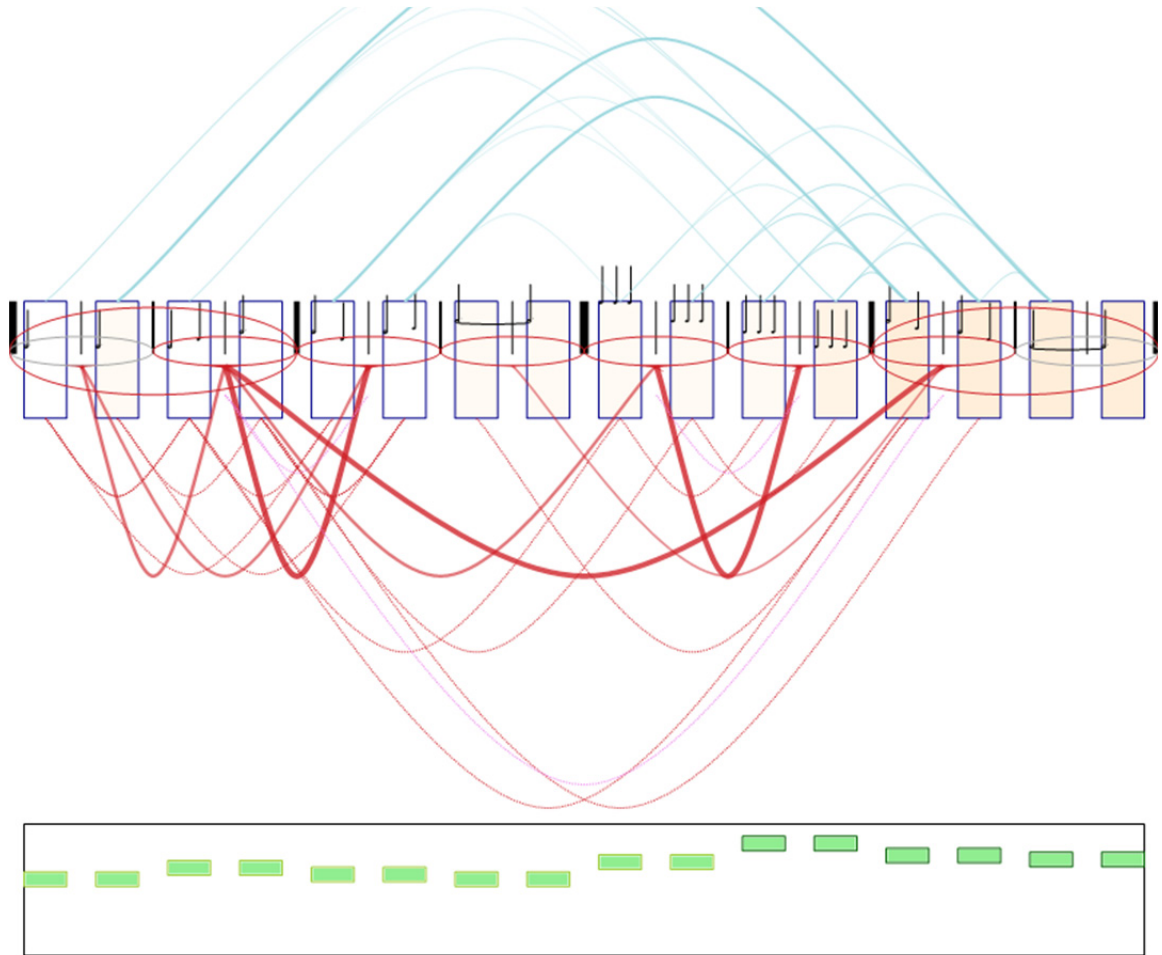


Figure 6.15: Row, Row, Row Your Boat (run 4).

This fourth run is also similar to previous runs, with a slightly different collection of analogies in place at the end of the run. This particular run gives a good picture of the types of relationships typically heard by the program, and, I believe, sheds some light on how people may hear this melody. I have already discussed most of the analogies present, but for a moment, reconsider the initial analogy $(1-2) \leftrightarrow (3-4)$, as well as the three strongest analogies in the figure above: $(3-4) \leftrightarrow (5-6)$, $(9-10) \leftrightarrow (11-12)$, and $(3-4) \leftrightarrow (12-13)$. Taken together, these four analogies tell a nice story and are evidence of a good listening performance

The initial analogy indicates that we hear a connection between the first and the second pairs of measures. (The grouping structure also indicates another type of connection; each of these measure pairs, 1–2 and 3–4, forms a connected unit, but in this discussion I focus on the analogies rather than the groups created.) Already, this contrasts greatly with the way the program listened to the random Bad Melody #1, in which no strong analogies or groups were heard.

The second analogy, (3–4)↔(5–6), indicates that the third small group (the third pair of measures) was heard as similar to the second, and the strength of the analogy shows that this relationship was quite salient. Now, it is important to notice what has happened: group (1–2) was mapped onto (3–4), which was itself mapped forward onto (5–6). This chain of two analogies linking three groups is striking in how it indicates the cohesiveness of the components in this melody, even though as the melody proceeds from measure 1 through 6, the rhythmic speed is increasing. Notice how different group (1–2) appears from (5–6). No *direct* analogy has formed between these two groups in this run (a weak analogy between them does form in run 5, below), but they are nevertheless linked indirectly through the analogy-chain, with the intermediate group (3–4) acting as a stepping-stone.

But what is the significance of this little chain of analogies? I claim that, taken together, these two analogies indicate one of the critical features of this melody that make it sound more *melodic* than the Bad Melodies earlier in this chapter: it has been constructed based on the principle of motivic development. Arnold Schönberg is best known as the father of *atonal* music, but he also taught *tonal* music theory and composition. His book *Fundamentals of Musical Composition* (1967) instructs students to build melodies by beginning with a short motive and developing it through a series of transformations as the melody moves forward. His book catalogs many types of such transformations and provides

many examples of how famous classical melodies can be heard in terms of these transformations. The little chain of analogies $(1-2) \leftrightarrow (3-4) \leftrightarrow (5-6)$ made by Musicat on this run shows that it sometimes hears these six measures in terms of this sort of motivic development, and this analysis is consistent with how Schoenberg suggests that we understand simple melodies.

Continuing to an analogy later in the melody, we find $(9-10) \leftrightarrow (11-12)$, which I discussed earlier. Notice how, in this hearing, the first six measures are linked together, followed by two measures which (unfortunately) were not heard as part of a larger group. Then, starting in measure 9, we encounter new material, and this analogy indicates how all four of these measures, 9–12, are heard in a linked-together manner.

The final strong analogy, $(3-4) \leftrightarrow (12-13)$, also was discussed earlier. But in this hearing it is particularly instructive to see how, after encountering the contrasting melodic material in measures 9–12, suddenly the program has heard a return to an earlier theme. Measures 12–13 remind the program of measures 3–4, and then the melody ends on a long note (although unfortunately, in this hearing that final note was not linked to something earlier, as it was in run 1).

This run was lacking some elements, but overall it is indicative of a quite reasonable overall way of hearing the melody. The perceived analogies have done a relatively good job of indicating how different melodic ideas are arranged and linked together within the melody. This is also an appropriate moment to remind the reader to consider what the fundamental purpose of the program is. I just stated that this run was “lacking some elements”, but our goal is not to model a “perfect” listening performance, as such a thing does not make sense. Human attention and perception are resource-limited and we can only attend to a certain number of things at once and they will not always be the same ones. Musicat’s explicit perception of an analogy in one run but not in another reflects this variable attention and the

uniqueness of each listening experience. (Of course, extremely salient features of a melody *should* be heard by the program on every run, and Musicat admittedly misses many very salient analogies and group boundaries that a good human listener would never miss.)

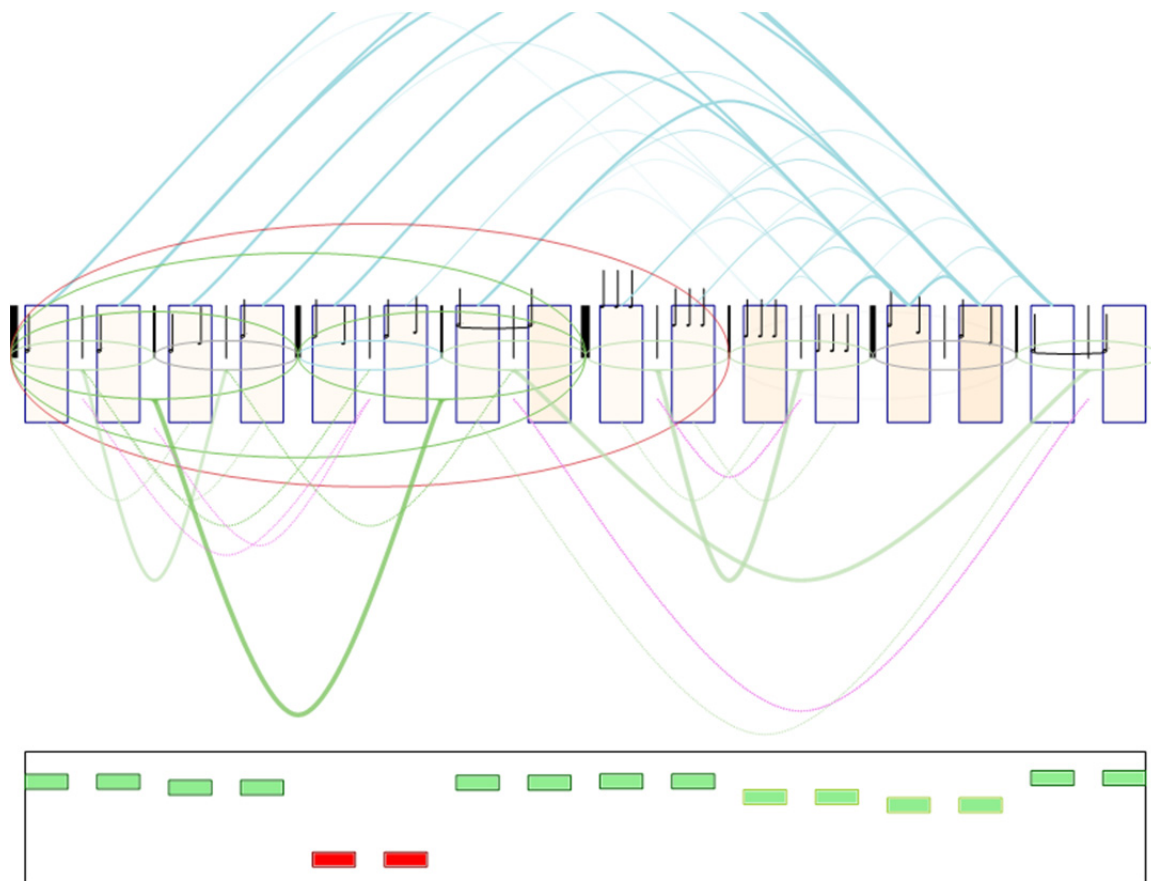


Figure 6.16: Row, Row, Row Your Boat (run 5).

In a final run on this melody, the grouping structure is different from that in earlier runs, and a larger analogy has formed than we have yet seen for this melody. A few of the familiar analogies from earlier runs are present, although they seem weaker in this run. Thus

I lowered the detail level to show more clearly which structures were strong in this run; see the next figure.

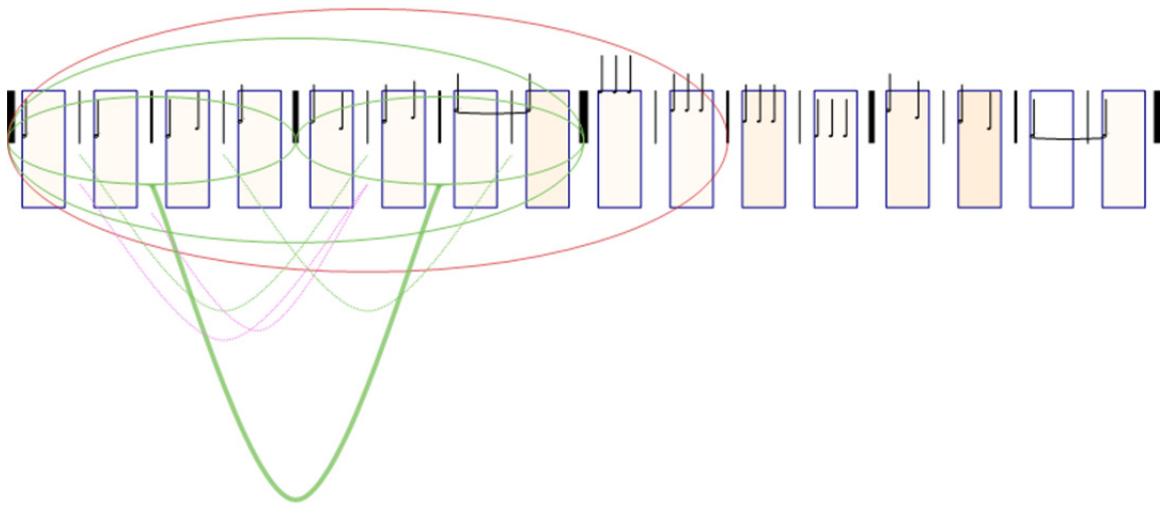


Figure 6.17: Row, Row, Row Your Boat (run 5), low detail level.

When the detail level for this run was reduced to “low”, many of the analogies and groups disappeared. The strongest analogy was the large one linking the first two 4-measure groups: (1–4)↔(5–8). It is nice to see that the program has heard all eight measures of the melody’s first half as forming a coherent unit (indicated by the green ellipse surrounding all of them), and it is consistent with my hearing that there is an analogy between the two 4-measure groups that make up this opening half of the full melody.

The red group (**1–10**), however, is a mystery, and it does not seem justifiable as a way of hearing the melody. In addition to this problem, all the smaller structures in the second half of the melody have disappeared at this detail level; they were all weak, indicating that the program did not create any strong structures in the second half.

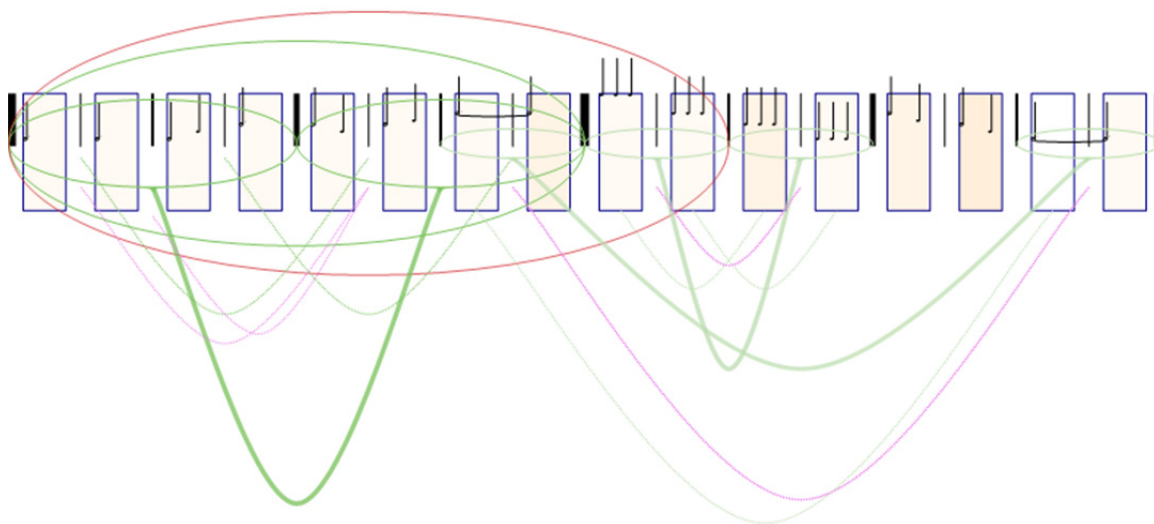


Figure 6.18: Row, Row, Row Your Boat (run 5), medium detail level.

I raised the detail level to “medium” in order to bring back into the picture some of the structures from the second half of the melody. The analogy $(9-10) \leftrightarrow (11-12)$ is visible again, although it is quite odd that the right-hand edge of the large red group cuts this analogy in half. The final note of the melody is now linked by analogy to the final note of the first half (as in run 1), which seems very reasonable. However, the descending line in measures 13–14 is not connected by any analogy to any earlier material.

This particular run was disappointing, but overall the program tended to find many of the expected analogies in this melody. Run 4 was particularly “good” in that it corresponded well with my own experience of listening to the melody. While the foregoing snapshots of these five runs are no substitute for seeing the program in action, I think that, taken together, they give a balanced picture of what Musicat perceives when it listens to “Row, Row, Row Your Boat”.

SUR LE PONT D'AVIGNON

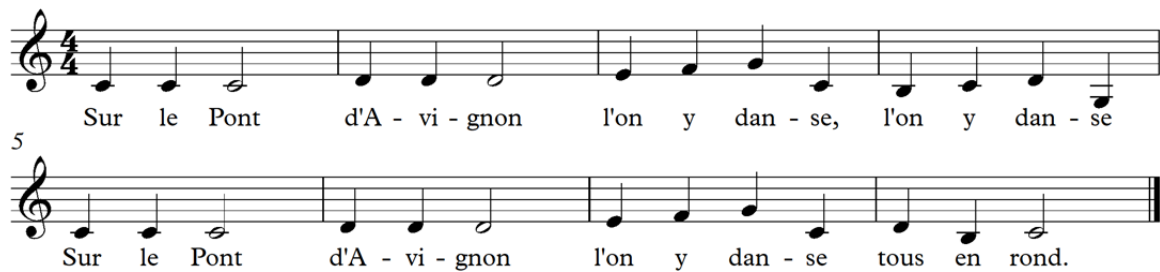


Figure 6.19: Sur le pont d'Avignon.

This short French children's song has a simple structure that is visible just by looking at the music in the figure above. The first four measures are aligned above the second four measures, making it easy to see that the two halves of the melody are nearly identical. The only difference is in measures 4 and 8. Measure 4 ends with a half-cadence on an implied G major chord (the dominant chord in C major), whereas measure 8 ends on C with an implied tonic chord. The program listens to melodies with no chords provided, but behind-the-scenes, it imagines chords implied by the melody, much as people do. These chords are not shown here, but they may help the program understand the melody, especially in obvious cases such as this one. A very simple implied harmony that we might imagine (with no prior knowledge of which chords are typically used to accompany this melody) is given below, with one chord per measure (except for the final measure, which has two chords):

C G C G C G C (G – C)

This melody is simpler than “Row, Row, Row Your Boat”, so I consider just two different listening performances below.

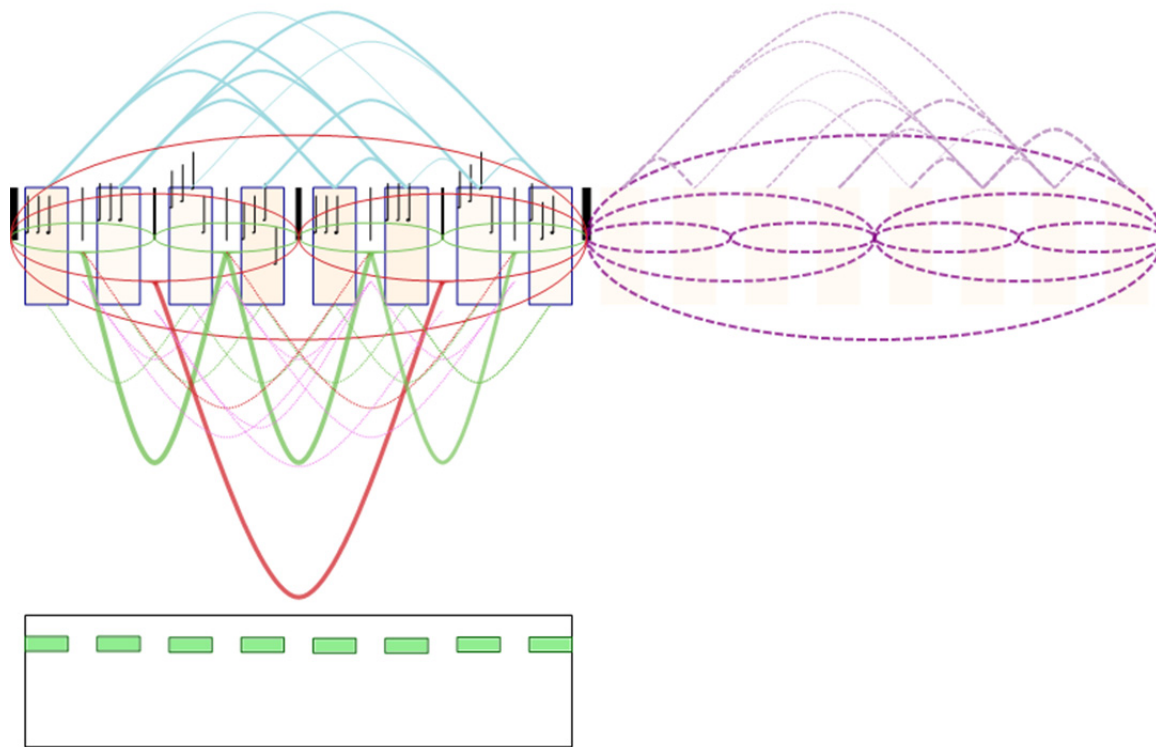


Figure 6.20: *Sur le pont d'Avignon* (run 1), just after hearing the final measure.

Musicat has heard the melody as having a completely regular binary grouping structure, and this grouping makes intuitive sense. Moreover, before the program realizes that the melody is over, it expects the grouping structure for the whole melody to be repeated exactly once again (shown in purple to the right).

To make the structure even clearer, I modified the detail level to produce the next figure (I also waited for the run to complete — Musicat continues “listening” for a little while after the last note arrives — but the picture did not change significantly).

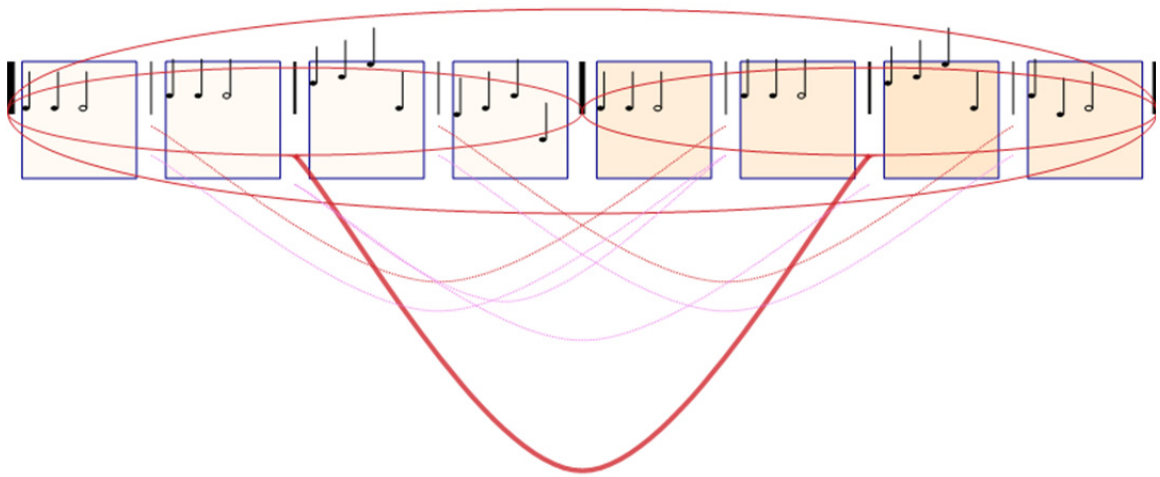


Figure 6.21: *Sur le pont d'Avignon* (run 1), low detail level.

This figure shows the strongest structures heard in this run. The melody has been divided into two nearly-identical halves — the program has noticed this obvious property of the melody, which I mentioned above — and a strong analogy (shown in red) has been made between the two halves. The analogy is supported by a variety of contour relationships shown in thinner lines in the figure.

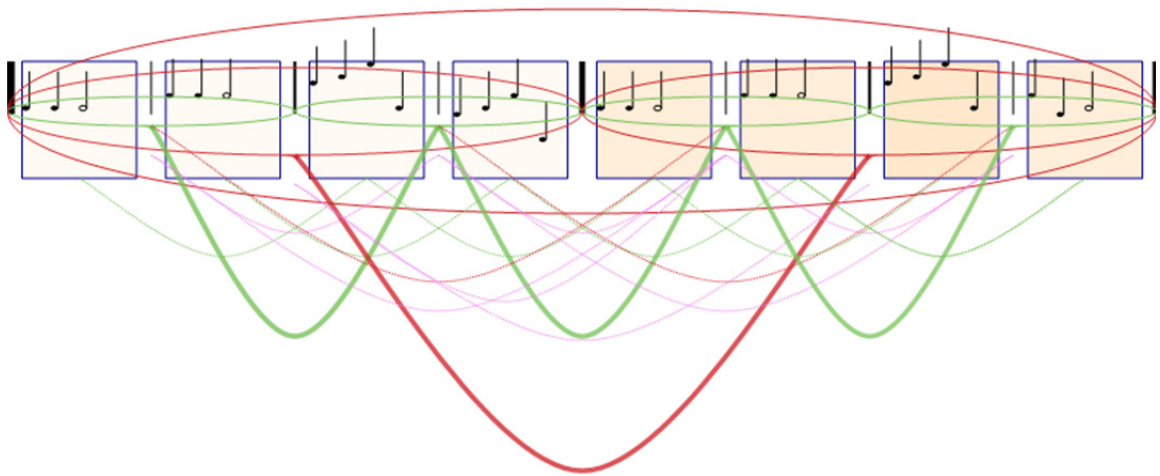


Figure 6.22: *Sur le pont d'Avignon* (run 1), medium detail level.

Moving the detail-level slider up to a medium setting makes some green analogies and groups visible. Each half of the melody is itself divided in half again (each red meta-group contains two smaller green two-measure groups).

Turning to the green analogies, we see two analogies that cover nearly the same material: (1–2)↔(3–4), and the very similar (5–6)↔(7–8). I say that these two analogies are very similar to each other because the two halves of the melody are almost identical. Why did the program form these analogies? It is easy to see that measure 1 is similar to measure 2 or that measure 3 is similar to measure 4, but the relationship between measures 1 and 3 or 2 and 4 is less obvious. The light green lines in the figure linking measure 1 to measure 3 and measure 2 to measure 4 indicate that Musicat found *rhythmic* relationships between these pairs of measures. Measure 1 differs rhythmically from measure 3 only by one quarter note, and the same holds for measures 2 and 4. Because rhythmic relationships are very salient for Musicat, it makes these analogies, even though the pitch contour of measures 1 and 2 is very different from that of measures 3 and 4.

The third analogy in the picture, (3–4)↔(5–6), actually follows the same logic. Indeed, because measures 5 and 6 are identical to measures 1 and 2, this is the same analogy as the two just discussed, except that the temporal order of the two groups involved has been reversed.

This series of three green analogies in a row is reminiscent of the analogies at the start of run 4 on “Row, Row, Row Your Boat”. However, they are less interesting to me in this case because instead of indicating a true motivic development in the melody, here they merely indicate that Musicat has noticed that the first two groups (1–2) and (3–4) are rhythmically similar, and that the whole first half was repeated. The larger red analogy in this picture seems more significant to me, and is more in tune with how I hear this melody.

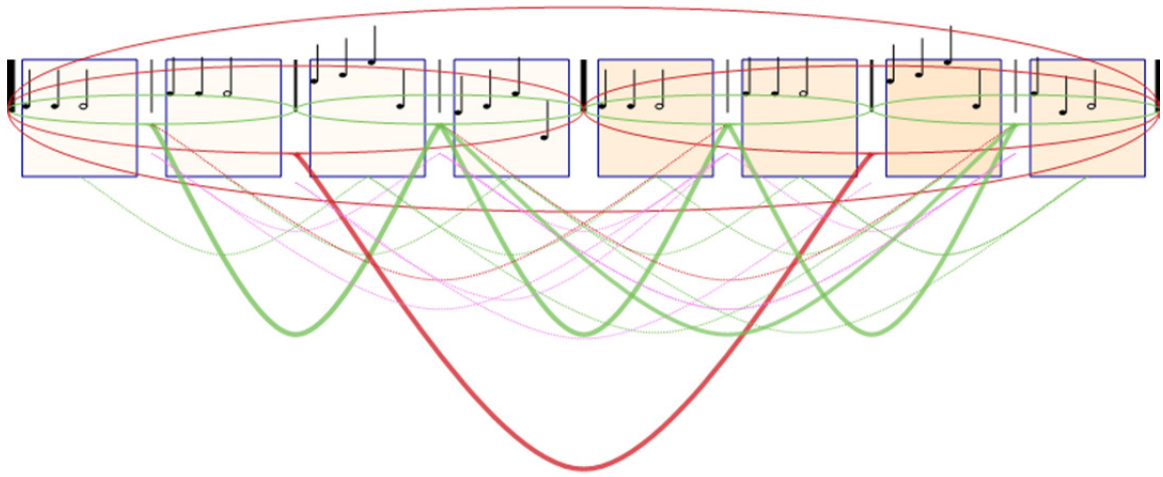


Figure 6.23: *Sur le pont d'Avignon* (run 1), high detail level.

Returning the detail slider to the “high” level, we notice that one more green analogy is visible: (3–4)↔(7–8). This analogy has formed even though the two groups involved end in slightly different ways. In this case, Musicat probably noticed that the left-hand group ended on the dominant note (and dominant chord), whereas the right-hand group ended on the tonic note and chord. The dominant→tonic relationship present in this green analogy most likely strengthened the large red analogy: Musicat is aware of this sort of simple dominant-to-tonic progression.

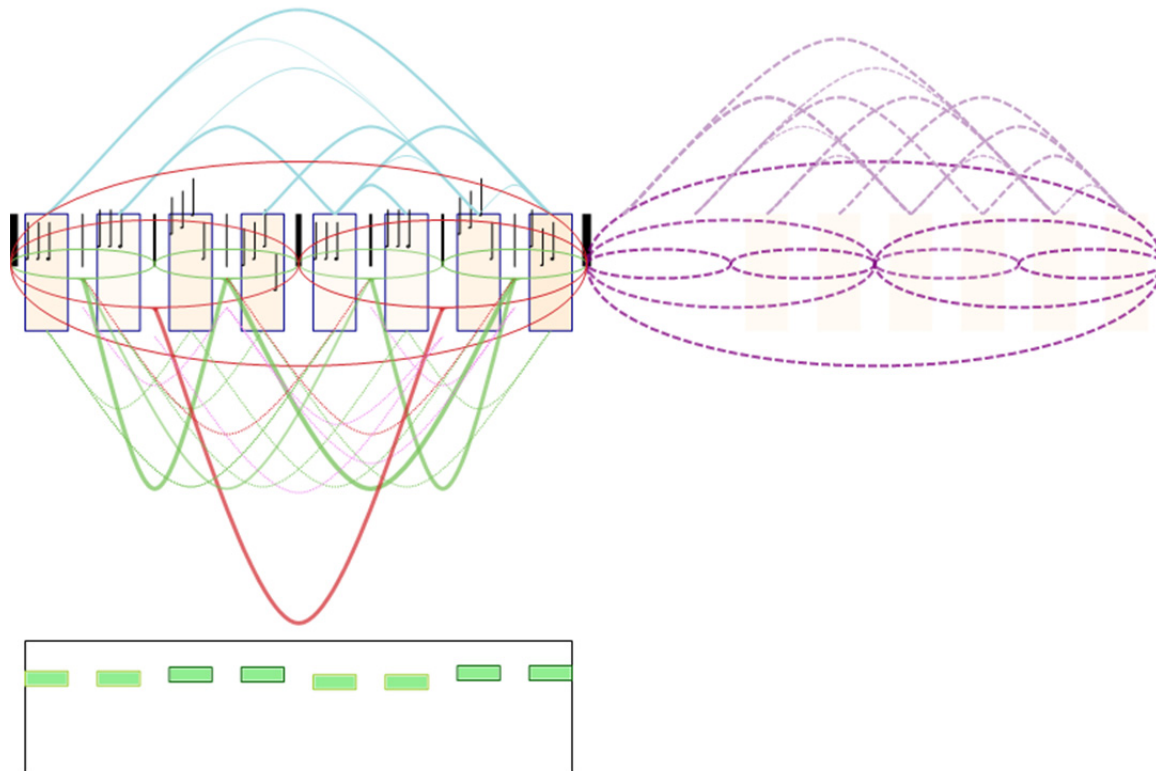


Figure 6.24: *Sur le pont d'Avignon* (run 2), just after hearing the final measure.

A second run on this melody resulted in a very similar picture. Two big differences here are that the middle analogy $(3-4) \leftrightarrow (5-6)$ is not present. Instead, the analogy $(1-2) \leftrightarrow (5-6)$ has formed (although it is quite weak at this point).

Recall that the program keeps listening for several more measures of time after the last note is heard. In this case the extra time made a difference in structure strength. The next figure shows the program's state at the conclusion of this extra listening time.

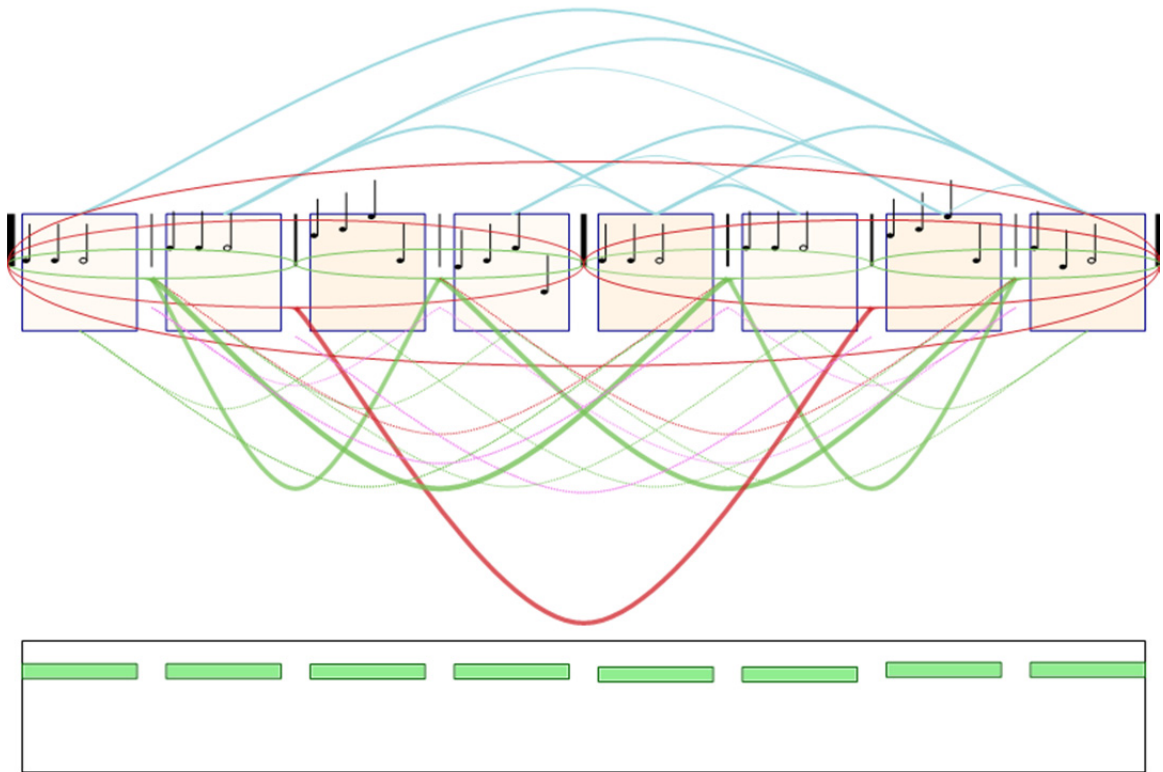


Figure 6.25: *Sur le pont d'Avignon* (run 2), final state.

Notice that the analogy $(1-2) \leftrightarrow (5-6)$ is stronger now, compared with the previous view. Indeed, the only weak structures are the blue measure links and the “next-door-neighbor” analogies $(1-2) \leftrightarrow (3-4)$ and $(5-6) \leftrightarrow (7-8)$. These disappear when the detail level is turned down, as is shown in the next figure.

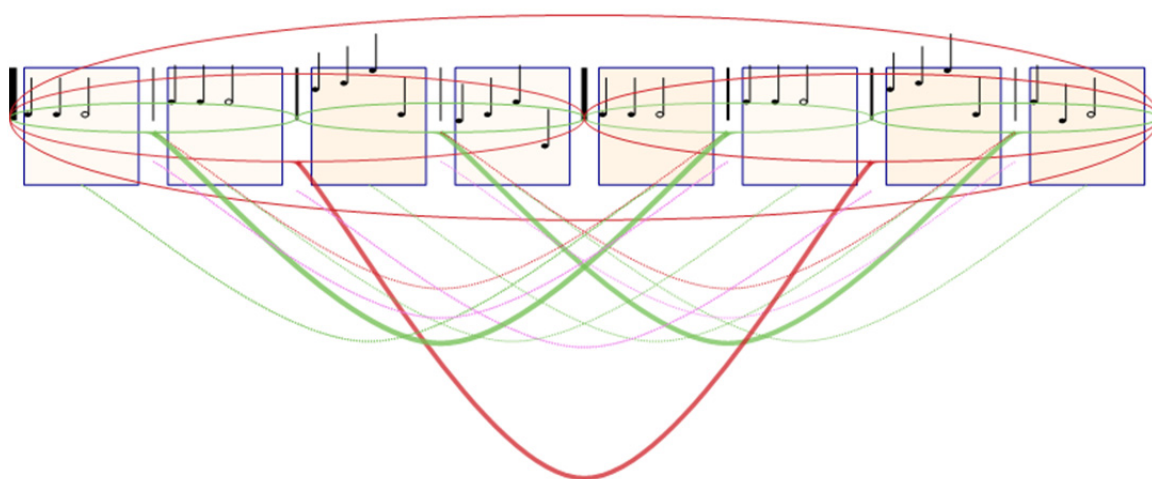


Figure 6.26: Sur le pont d'Avignon (run 2), low detail level.

At a low detail level, only the two strong long-distance green analogies and the large red analogy remain. Again, notice how much this situation contrasts with the one earlier involving the Bad Melodies, in which nearly all structures disappeared at low detail.

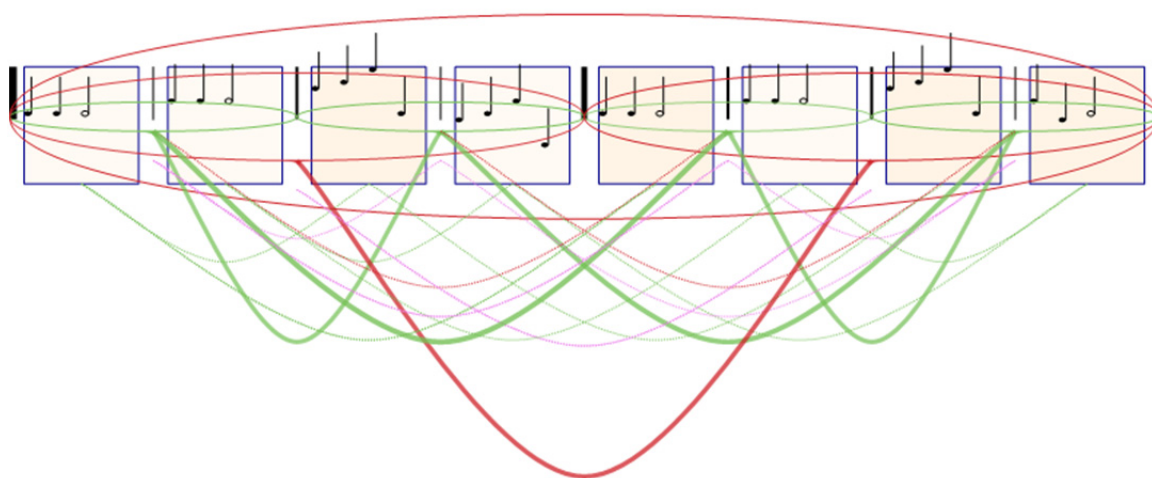


Figure 6.27: Sur le pont d'Avignon (run 2), medium detail level.

This medium-detail view shows the important structures formed in this run. I think that this listening performance was slightly better than the previous run; the long-distance analogies (1–2)↔(5–6) and (3–4)↔(7–8) are salient and important to a human listener.

Even though this run seems like a good performance, Musicat still missed some important features of this melody that human listeners would likely notice. Most significantly, there are stepwise relationships between measures that are not explicitly noticed by the program, but which contribute significantly to the smoothness and singability of the melody line. For instance, the first measure consists of three C's while the second measure has three D's. The program can recognize these measures as having the same rhythm and pitch contour, but it would not affect Musicat's hearing much if each D in the second measure were replaced with E or even F. The relationship of a single scale step between measures 1 and 2, however, helps a *human* listener to group the two measures together. More importantly, the interesting melodic pattern in measure 3 is repeated, with transposition, in measure 4. Musicat may notice this transposition and repetition (although it does not currently display these things with any special notation, such as the blue zigzaggy lines used to indicate sequences of three or more items related by transposition), and when they are noticed, Musicat is more likely to form a group consisting of two measures involved. However, the program would not notice the smooth stepwise connection between the final note of measure 3 (C) and the first note of measure 4 (B). A stepwise connection is also present at the end of the melody between measures 7 and 8, although in this case the C is connected to the D one step above. These types of melodic connections are very important in human listening, but unfortunately the present version of Musicat does not notice them.

FRÈRE JACQUES



Frè - re Jac - ques, frè - re Jac - ques, dor - mez vous? Dor - mez vous?

5 Son-nez les mat - i - nes! Son-nez les mat - i - nes! Din, dan, don. Din, dan, don.

The musical score for 'Frère Jacques' is written on two staves. The first staff contains the first four measures of the melody, which are repeated. The second staff, starting at measure 5, contains the next four measures, which are also repeated. The lyrics are written below the notes.

Figure 6.28: Frère Jacques.

Our next example melody is another French folk tune (but this one also is well known to English speakers: “Are You Sleeping, Brother John?”). A glance at the music above reveals that it also has a simple structure, but this structure is different from that of “Sur le pont d’Avignon”. In that melody, the second half was a nearly exact repetition of the first half. In “Frère Jacques”, however, every single measure is followed by an exact repetition. Instead of having a block of four measures repeated one time, we have four single measures, each immediately repeated. To make this point clear, I re-composed “Frère Jacques” to follow the formal pattern of “Sur le pont d’Avignon”:



Frè - re Jac - ques d'A - vi - gnon, Son-nez les mat - i - nes! Din, dan, don.

5 Frè - re Jac - ques d'A - vi - gnon, Son-nez les mat - i - nes! Din, dan, don.

The musical score for 'Frère Jacques d'Avignon' is written on two staves. The first staff contains the first four measures of the melody, which are repeated. The second staff, starting at measure 5, contains the next four measures, which are also repeated. The lyrics are written below the notes.

Figure 6.29: Frère Jacques d’Avignon.

These two melodies look and sound quite different. How will Musicat’s listening performance on Frère Jacques differ from its perception of “Sur le pont d’Avignon” in the

previous section? (Incidentally, I have not run Musicat on the re-composed melody, although I would guess it would hear it in a very similar way to the way it heard “Sur le pont d’Avignon”.)

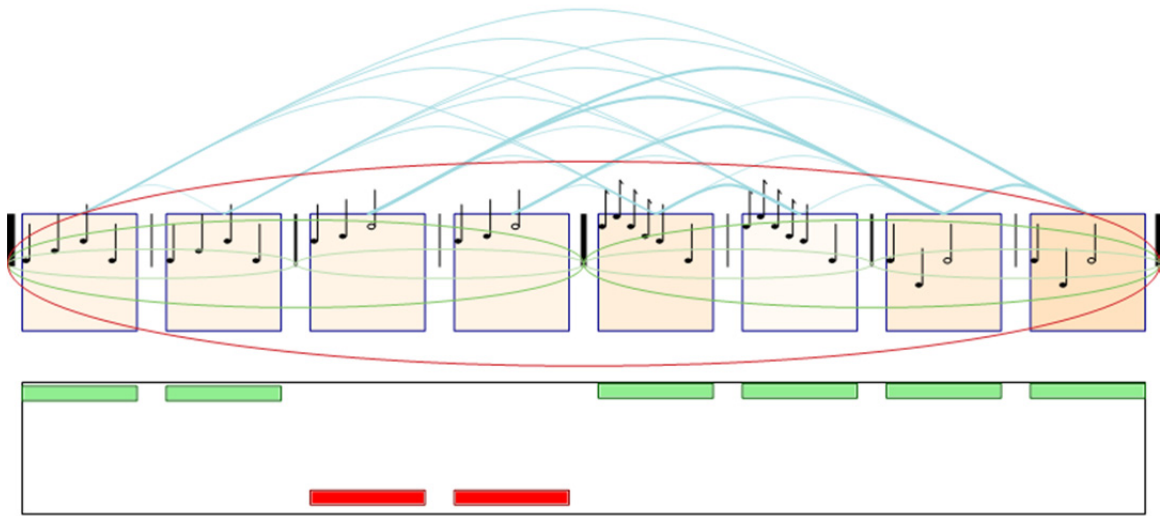


Figure 6.30: Frère Jacques (run 1).

This run produced the same grouping structure as we saw in “Sur le pont d’Avignon”, but no analogies were found. Musicat does indeed hear these melodies in a different way (at least in this run).

Although no analogies were found in this run, it is worth reminding the reader that the blue lines above the figure also indicate relationships between measures, and we can certainly call these “analogies” as well. For technical reasons described in Chapter 9, these simple relationships between measures, based solely on rhythmic similarity, are handled separately from the more complex analogies drawn below the measures. However, in this melody there is, of course, plenty of analogy-making at work, as always (recall the discussion of the pervasiveness of analogy in Chapter 4).

Because this melody consists of exact repetitions of each measure, the most obvious analogies are the relationships between adjacent measures. Analogies drawn below the

measures in Musicat usually span more than two measures, but the blue measure links above show that Musicat has noticed the rhythmic repetition. For instance, the blue arc between measures 5 and 6, just like the one between measures 7 and 8, indicates exact rhythmic repetition. Musicat almost certainly created arcs such as these between measures 1 and 2 and between measures 3 and 4 during the run, but by the end of the run they have faded away. Those arcs, however, likely helped to create all the groups consisting of measure pairs in the melody: (1–2), (3–4), (5–6), and (7–8). In summary, these repetitions were not unnoticed by the program, even though that fact is not obvious in the figure above showing the program's final state.

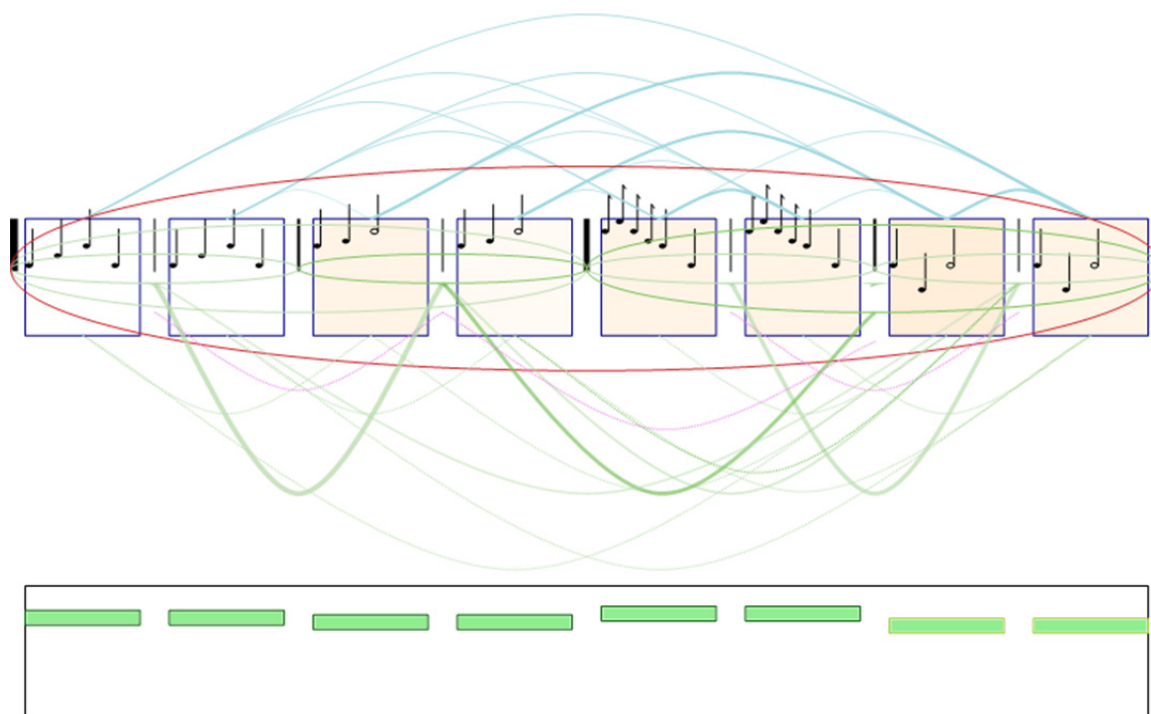


Figure 6.31: Frère Jacques (run 2).

A second run resulted in the same *grouping* structure, but in addition to groups we now have three green *analogies*. This picture is slightly reminiscent of the pictures of “Sur le pont d’Avignon”, but there is no large red analogy linking the first half of the melody to the second half. Also, two of the green analogies are weak. The next figure uses the detail level to highlight the strongest structures:

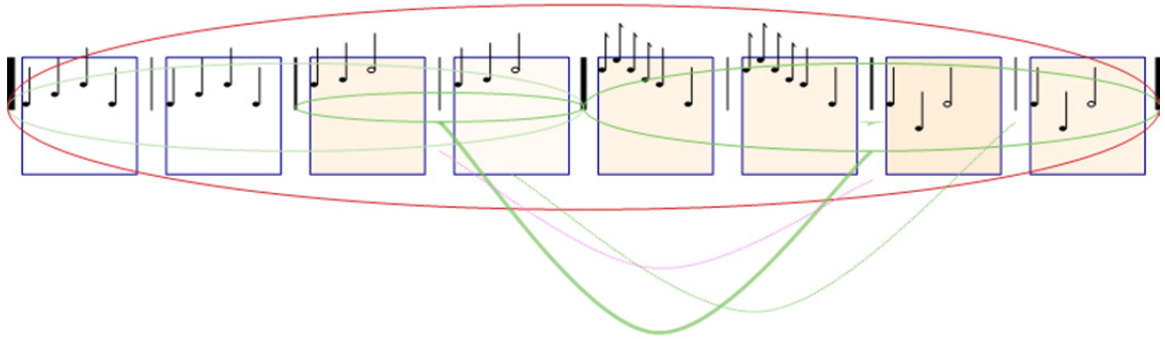


Figure 6.32: Frère Jacques (run 2), strongest structures.

At this low detail level, only one green analogy remains, and some of the groups have also disappeared. Notice that this is a surprising analogy, because it is between differently-sized structures: $(3-4) \leftrightarrow (5-8)$. The thin green arc indicates that the analogy is partly motivated by a rhythmic relationship: the two measures 3–4 have the same rhythm as 7–8. The pink arc indicates some sort of non-rhythmic relationship, which most likely is the dominant→tonic relationship implied by the left-hand group (3–4) ending on the dominant note G and the right-hand group (7–8) ending on the tonic, C.

This run, like the previous one, resulted in different behavior from runs on “Sur le pont d’Avignon”. However, because some of the analogies were similar in both performances, I ran it one more time to verify that Musicat hears these two melodies in a significantly different way.

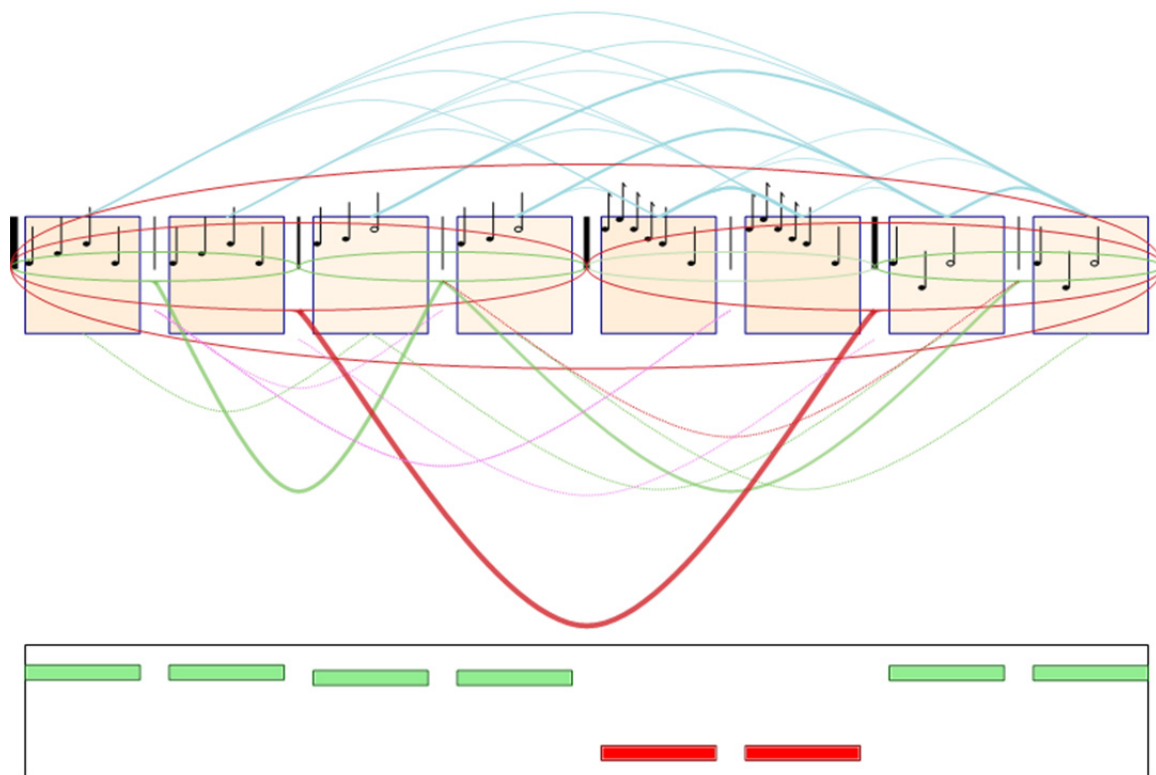


Figure 6.33: Frère Jacques (run 3).

On the third run, a large red analogy formed, and again we have the same group structure as before. Did the program hear this melody as if it were equivalent to “Sur le pont d’Avignon” after all? Or are the analogies formed here still reasonable ways of hearing musical similarity, even though the melody is not constructed as a large-scale repetition of four initial measures? The analogy (3–4)↔(7–8), at least, looks very reasonable. I lowered the detail level to remove the green analogies from the picture so I could investigate the red analogy more closely.

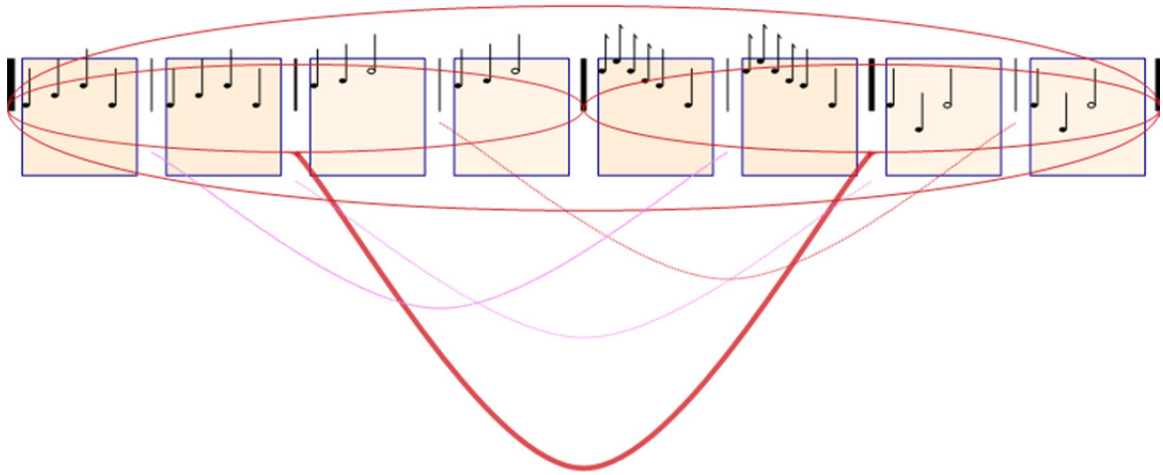


Figure 6.34: Frère Jacques (run 3), low detail.

This analogy connects the two halves of the melody, just as in “Sur le pont d’Avignon”, run 1. However, there are far fewer supporting components of this analogy in this run on “Frère Jacques” than there were in that run on “Sur le pont d’Avignon”: there are just two pink arcs (left and center), indicating non-rhythmic relationships, and one light red arc (right) indicating a rhythmic relationship. This rhythmic relationship is the same one discussed in run 2; that is, both halves of this melody end with the rhythm QQH QQH. The possible domaint→tonic relationship between the two groups was also mentioned above, and it may be part of the analogy here (unfortunately, although these details are available to the user of the program while it is running, they are not visible in the pictures). Another possible source of this relationship is a contour relationship between the first two measures of each half of the melody — that is, between measures 1–2 and measures 5–6. Specifically, each of measures 1 and 2 has a rising and then falling contour, in which the upwards motion is by *step* and the downwards motion is by *leap*. Measures 5–6 have a different number of notes but the contour is related to that of measures 1–2: there is a one-note stepwise ascent followed by a stepwise descent and finally a downwards leap. Musicat considers these

contours to be just slightly similar; human listeners may even hear them as more similar than the program does.

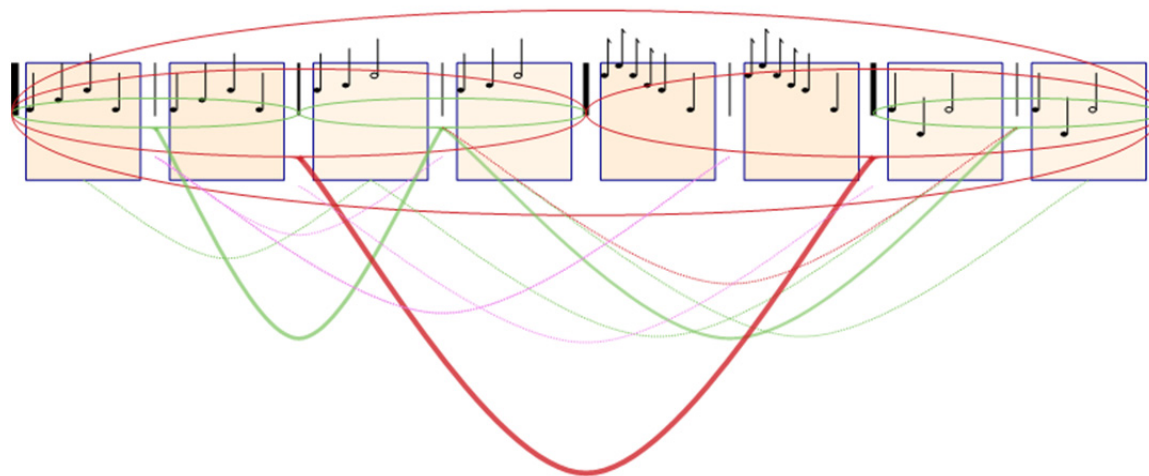


Figure 6.35: Frère Jacques (run 3), medium detail.

This figure shows the final state with the detail level turned back up to “medium”. Turning our attention to the first green analogy, $(1-2) \leftrightarrow (3-4)$, we see that this analogy looks similar to the first analogy in “Sur le pont d’Avignon”. Perhaps surprisingly, the first four measures of “Frère Jacques” are quite similar to the first four of “Sur le pont d’Avignon”. In both melodies, measure 1 has the same rhythm as measure 2. Likewise, measures 3 and 4 share a rhythm. In “Frère Jacques” the first rhythm is four quarter notes (QQQQ), followed by the exact repetition, and then QQH (and another QQH). In “Sur le pont d’Avignon”, these same two rhythms are also involved, just in a different order: QQH QQH QQQQ QQQQ. “Frère Jacques” features exact pitch repetition, while “Sur le pont d’Avignon” uses transposition of the pitches in the melody, but this is a minor difference. It is not surprising, then, that this large-scale analogy is found by Musicat in both melodies.

In sum, this listening performance by Musicat indeed makes sense; perhaps these two melodies are more similar than it first appears. The nature of the large red analogy, however, is quite different for the two melodies. In “Sur le pont d’Avignon”, the analogy was due to a near-exact repetition of the two melody halves. In this run of “Frère Jacques”, on the other hand, it was due to a few different factors, including rhythmic repetition and contour similarity.

SICILIENNE (FAURÉ)



Figure 6.36: Fauré’s Sicilienne for cello and piano, op. 78.

The first four measures of this short excerpt from the Sicilienne for cello and piano by Gabriel Fauré look very similar to the last four measures, just as was the case with “Sur le pont d’Avignon”. Thus we would expect to hear the strong analogy (1–4)↔(5–8) in this melody. However, the analogy is slightly subtler here: in “Sur le pont d’Avignon”, the first three measures of the first half were exactly the same as the corresponding measures in the second half. In this Sicilienne, however, the two halves have the same rhythm but different pitches. The melody *contour*, however, is the same in the two halves. Will Musicat recognize the similarities?

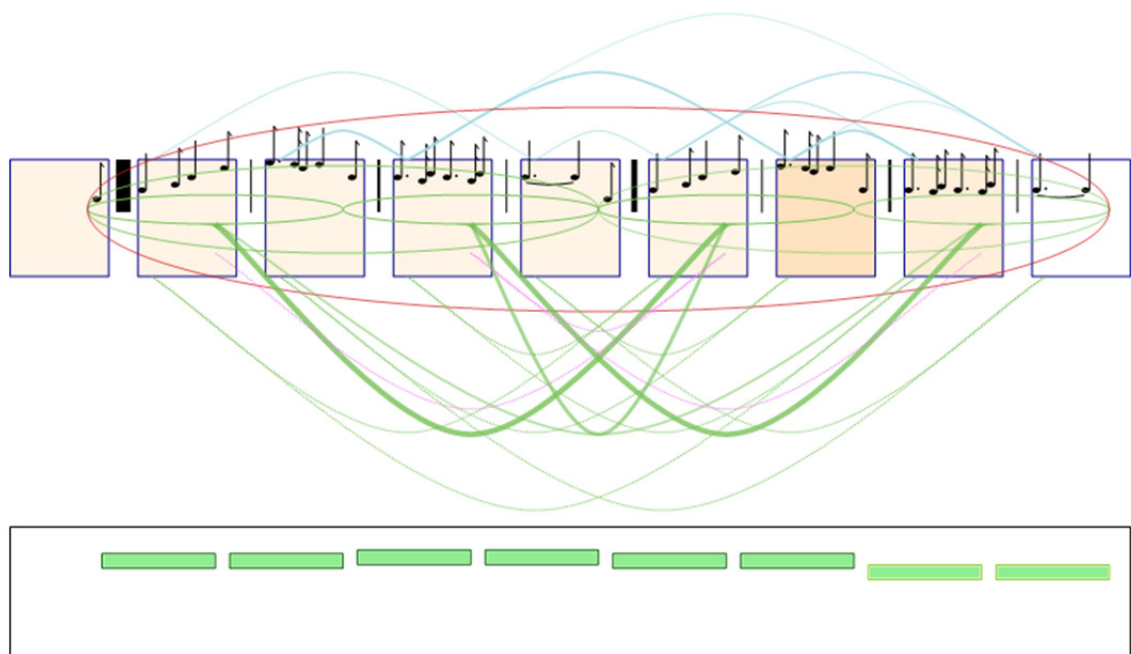


Figure 6.37: Sicilienne (run 1).

In this first run, the program *almost* found the analogy I expected. In the final output (Figure 37), there is no large-scale analogy linking the two halves of the melody, but a reasonable grouping structure has appeared, and all the components of the two halves have formed the expected analogies. That is, (1–2) was mapped onto (5–6), and (3–4) was mapped onto (7–8). Additionally, the weaker analogy (3–4)↔(5–6) has formed because of some perceived rhythmic similarities.

In the previous paragraph, I stretched my notation in order to accommodate the presence of upbeats (or pickup notes) in the melody: measure “1” does not indicate the first full measure after the thick bar line, as one might expect, but rather it stands for the first *measure-length segment* of the melody line (*i.e.*, in this case, the first 6-eighth-notes time span in the melody, starting on the low eighth note E and ending just after the quarter note E an octave higher). Likewise, groups always start and end just before final eighth-note upbeats

(groups are composed of measures and must also accommodate upbeats.) In the figure above, the ellipses representing groups include the pickup notes and hence are shifted to the left.

That Musicat accommodates the upbeats in this melody should not be seen as a sign of sophistication; to the contrary, it forces groups to include upbeats in a completely mechanical way. For melodies having an upbeat at their start, such as this one, Musicat simply assumes that all melodic phrases it hears will have a regular structure in which every group formed will start or end just before the metric position indicated by the first upbeat. Also recall that groups in Musicat always have a duration that is an integral multiple of the measure length. A more flexible version of the program would relax these constraints and would allow group boundaries to occur anywhere within measures (this would also be necessary for the program to make sense of melodies having internal time-signature changes). Chapter 8 has a discussion of a previous version of Musicat that did feature this flexibility in grouping.

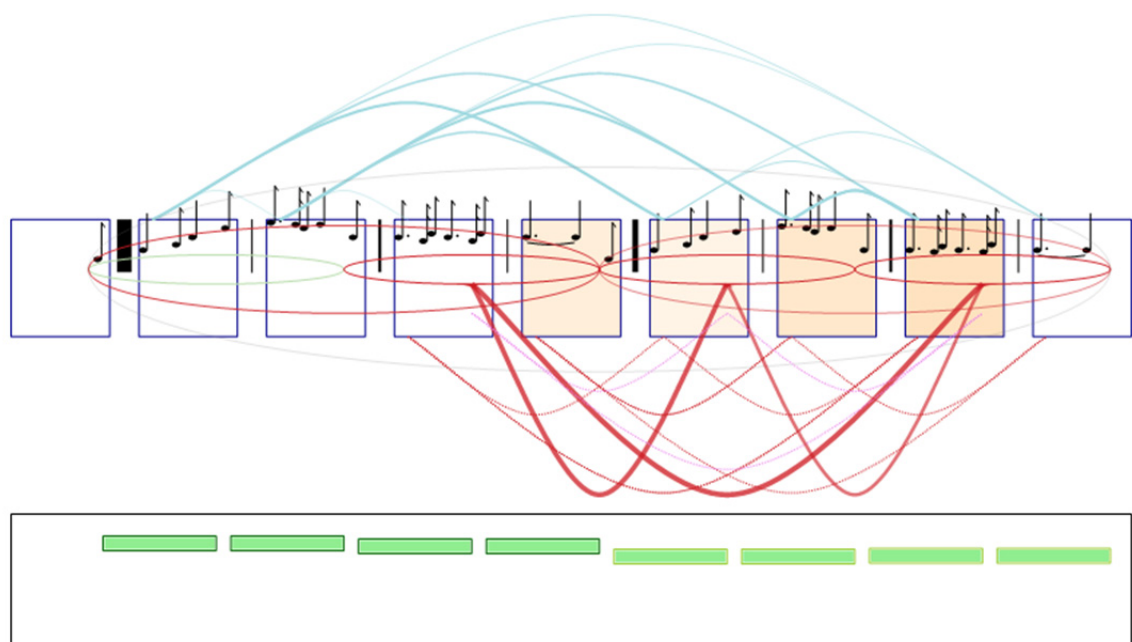


Figure 6.38: Sicilienne (run 2), final state.

At the very end of a second run, the picture is similar (Figure 38). Analogy $(1-2) \leftrightarrow (5-6)$ did not form this time, although a different analogy, $(5-6) \leftrightarrow (7-8)$, formed. The desired big-picture analogy, $(1-4) \leftrightarrow (5-8)$, however, is not visible. But let's take a look at the final stages of the run before it completely stopped listening:

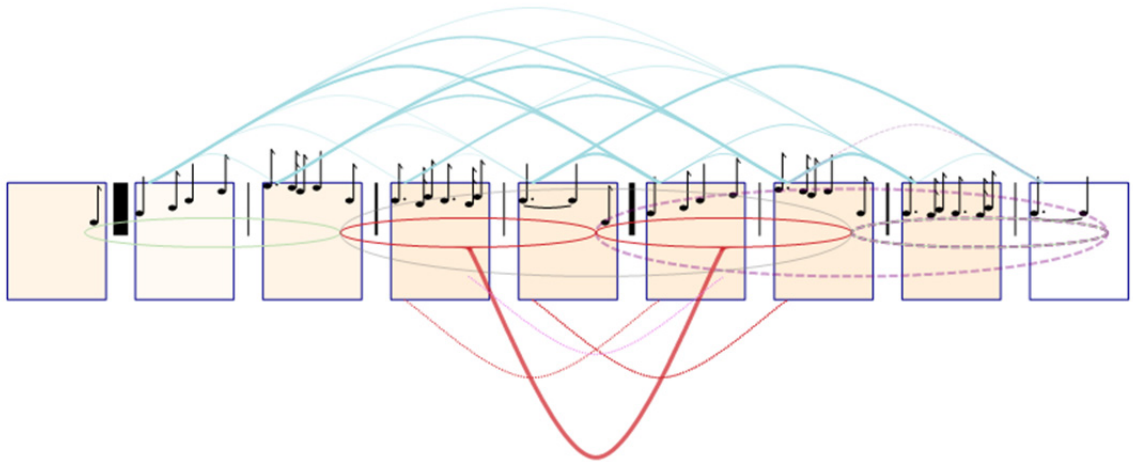


Figure 6.39: *Sicilienne* (run 2), measure 8(+1).

After one measure of “post-melody” listening time had passed, only one (very weak) analogy was visible...

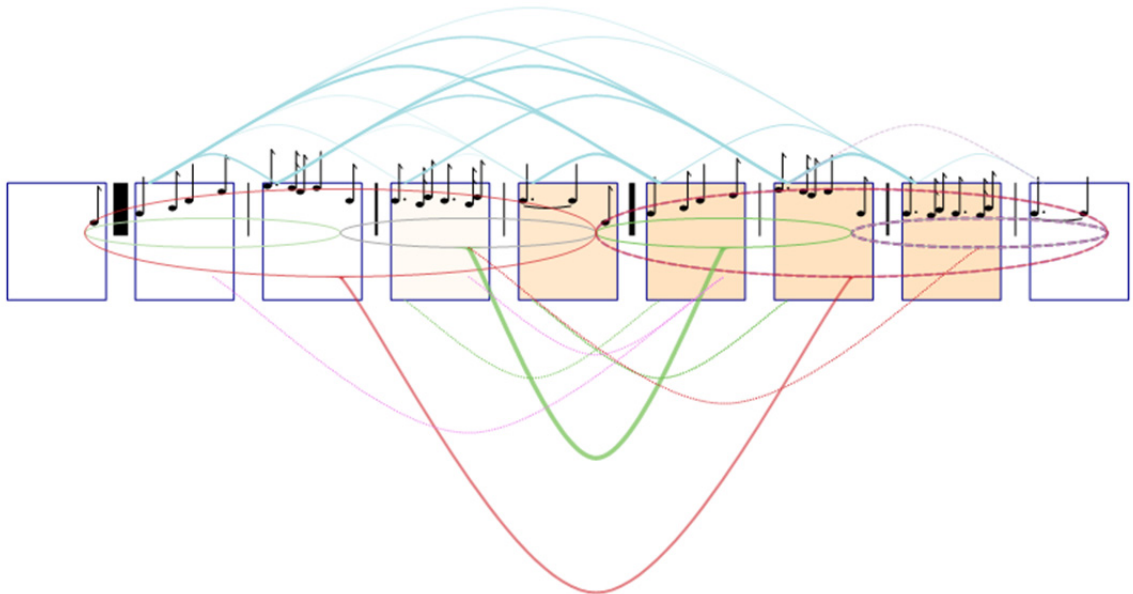


Figure 6.40: *Sicilienne* (run 2), measure 8(+2).

...but after one more measure, our desired large-scale analogy was formed: (1–4)↔(5–8), colored red in the figure.

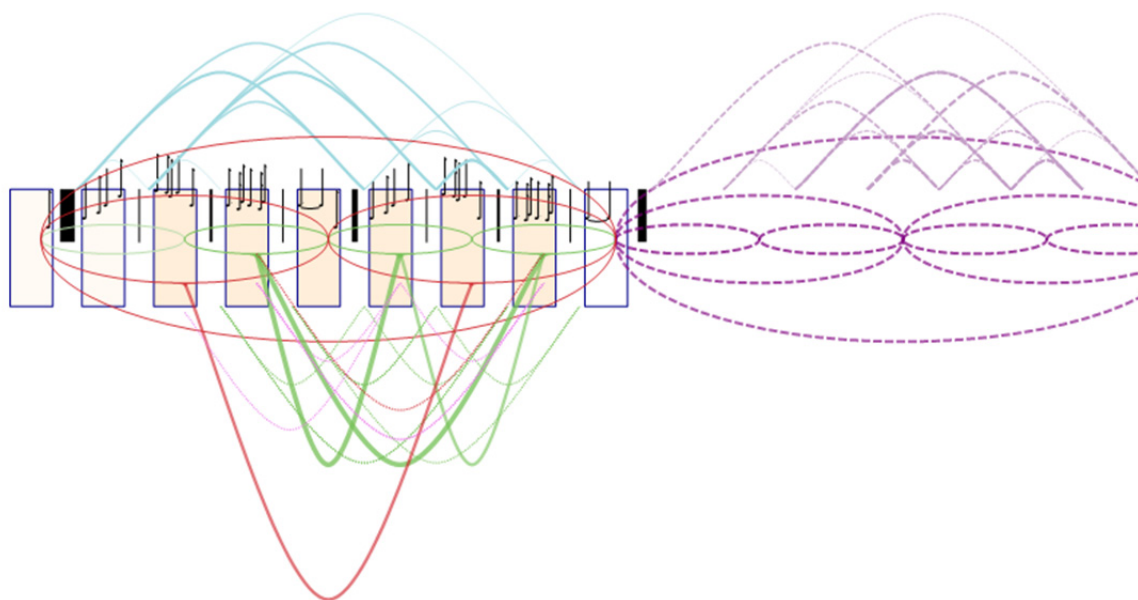


Figure 6.41: *Sicilienne* (run 2), measure 8(+5).

After a few more post-melody measures, more green analogies had formed, and the program had generated a large-scale purple expectation. However, in the program's final state, just moments later, the big red analogy suddenly disappeared:

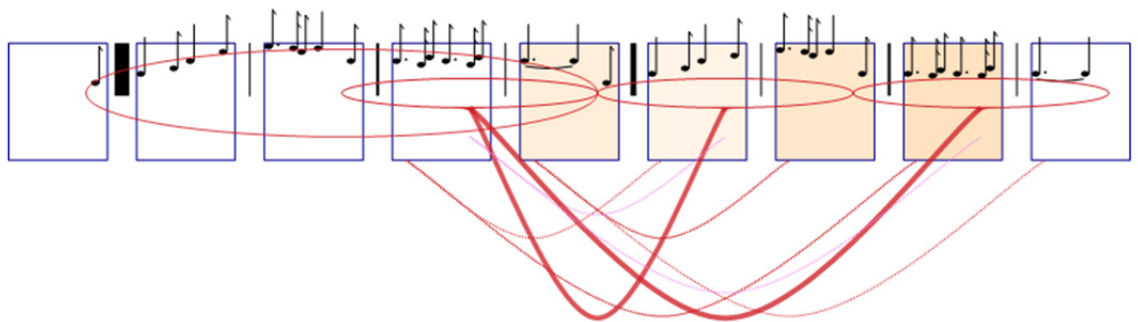


Figure 6.42: *Sicilienne* (run 2), final state, medium detail.

Why did the big analogy disappear? Viewing the final state at a medium detail level (as well as taking a closer look at the figure that included the big analogy) yields an

explanation: the root cause was that group (1–2) was too weak. Even though the group was formed, it was still weak (notice its light color in the high-detail figures earlier). Because (1–2) was weak, analogy (1–2)↔(5–6) did not form during this run. In Musicat, an analogy is strongest when all of its *components* (all the groups on its left- and right-hand sides) are involved in the mapping. In the figure above, where the large analogy did form momentarily, we can see that the group (1–2) was not involved in the mapping, and hence the big analogy was too weak and was destroyed.

One of the behind-the-scenes difficulties in this case is that group (1–2) must be formed and must achieve a high-enough strength value fairly early in the run. Musicat, remember, focuses its perceptual energy on the most recent several measures of music that it has heard. If, as in this example, a structure formed at the start of a melody is not strong enough, it is possible that the structure will remain weak when it eventually moves out of the simulated working-memory area (“the mind’s ear”). After that, the program cannot go back and re-hear the structure. In this example, group (1–2) simply never was perceived as a strong group, and this weakness prevented the formation of the key small-scale analogy (1–2)↔(5–6), and finally, this missing analogy in turn resulted in a lack of evidence to support the large analogy (1–4)↔(5–8).

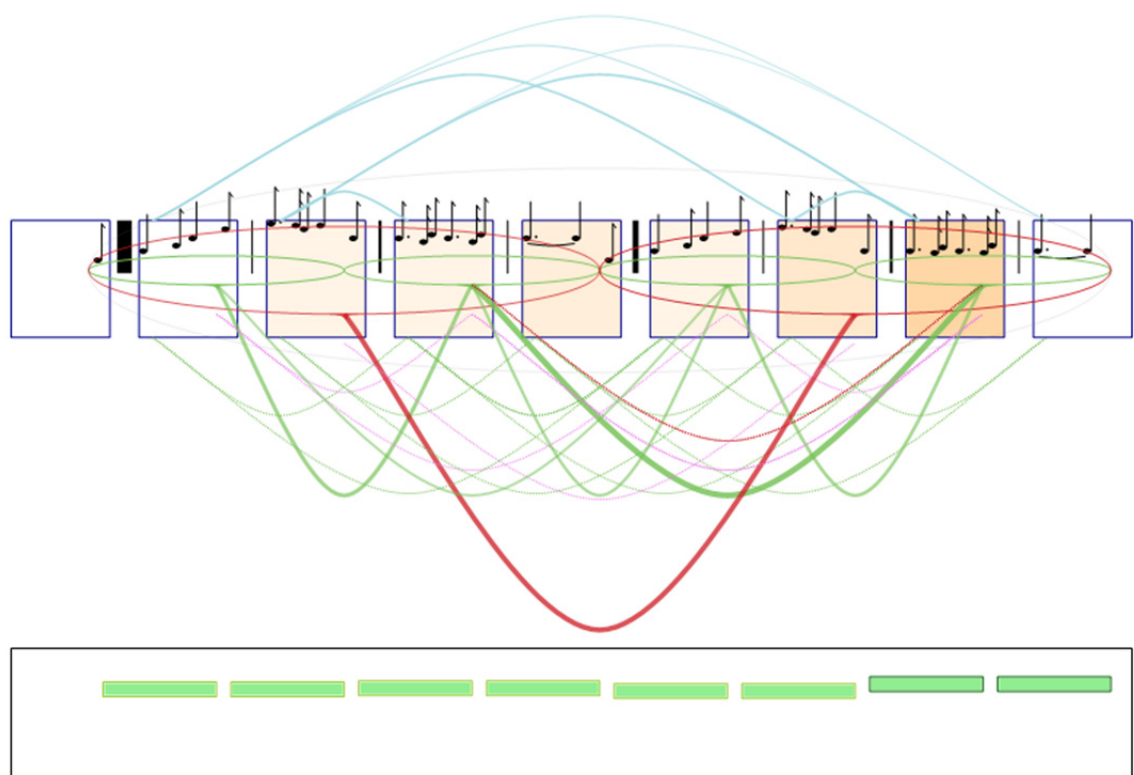


Figure 6.43: Sicilienne (run 3).

On a final run, Musicat *did* discover the big analogy between the two halves of the melody, and the analogy remained stable and strong all the way through the very end of the run. But even here, notice that the sub-analogy $(1-2) \leftrightarrow (5-6)$ is still much weaker than $(3-4) \leftrightarrow (7-8)$.

These three runs give a sense for the difficulty of Musicat's domain. To a human listener, the large analogy between the two halves of this melody is so obvious that it seems trivial. Even a non-musician looking at the music notation for this melody would be likely to recognize the similar shape formed by notes in the two halves of the melody. It is frustrating that Musicat still has trouble discovering and maintaining this analogy, but we can see that the analogy's formation depends on many substructures being perceived in a certain way.

It may be tempting to pessimistically imagine that Musicat's analogy-formation is akin to building a house of cards, vulnerable and at risk of collapsing if subjected to the slightest breeze. Musicat's analogies are *not* this fragile, fortunately: analogies, sub-analogies, and groups mutually reinforce each other, so that the whole structure can be stronger than the sum of its parts. However, the temporal nature of the domain complicates the picture, as we saw most clearly in run 2 above: if certain structures do not form quickly enough, their nonexistence (or weak existence) may prevent the formation of larger structures that are only perceivable later on. Even though this causes problems, as in run 2, where it was difficult for the program to create the large analogy, it is a natural consequence of the program's modeling of temporal perception. If a human listener, after all, were distracted during listening to the first few measures and failed to form a strong mental representation of the first half of the melody, we would not expect that person to notice an analogy between the two halves. Even the listening performances of the first two runs for this melody, then, may illustrate behavior that is "correct" in that it exhibits the expected sorts of missing large-scale structures after the end of the second half of the melody, given listening "mistakes" it made in early measures.

Even the simple-sounding melodies in this chapter posed formidable challenges to Musicat. The next chapter gives several examples of even more complex melodies, which will pose even more problems for Musicat. These will illustrate the kinds of groups and analogies it can make consistently, as well as indicate more areas for future improvement.

CHAPTER SEVEN

Musicat Listens to Complex Melodies

YOUNGER THAN SPRINGTIME (RODGERS AND HAMMERSTEIN)

5 Youn-ger than Spring-time___ are you, Sof-ter than star - light___ are you,

10 War-mer than winds of June are the gen-tle lips you gave me. Gay-er than laugh-ter

15 ___ are you, swee-ter than mu- sic___ are you. An-gel and lo-ver, hea-ven and Earth are

21 you to me. And when your youth and joy in - vade my arms, and fill my

27 heart as now they do, then, Youn-ger than Spring-time___ am I,

gay-er than laugh-ter___ am I. An-gel and lo-ver, hea-ven and Earth am I with you.

Figure 7.1: Younger than Springtime (from the musical *South Pacific*), 32-measure excerpt.

“Younger than Springtime”, with music by Richard Rodgers and lyrics by Oscar Hammerstein II, is our first example of a “Complex Melody”. In contrast to the short melodies of the previous two chapters, this is a typical song-length melody: 32 measures of music are included here (I omitted an introduction section that is part of the complete song). The length poses some challenges to Musicat (see the Chapter 10 for more discussion of this point), so in this section I have the program analyze a few individual shorter segments: measures 1–8, then 1–16, and finally measures 17–24. Measures 25–32 are extremely similar to 1–8 so I left them out of these runs.

It’s important to remember that the Musicat does not “hear” the lyrics — it is given only the notes. These lyrics provide many hints as to the grouping structure of the piece and suggest numerous analogies, but the program is working without these extra clues. For instance, the words “are you” at the end of each of the two phrase in “Younger than springtime are you / softer than starlight are you” help increase the feeling of parallelism between measures 1–2 and 3–4 (although the rhythmic and melodic similarity by itself is plenty enough for anyone to hear the analogy between the measures, without the lyrics). In the final 8 bars, moreover, the clever switch from “are you” to “am I” helps establish that this is the *final* section of the melody, wrapping up this 32-measure structure. In addition, the reprise of the words “younger than springtime” in measure 25 instantly indicates that this is the start of another section, related to the first one. It is thus almost trivial for a person who has access to the lyrics to hear that measure 25 is the start of a group. This final section is a minor variant, **A’**, of the first part of the melody, **A** (the large-scale structure can be heard as **AABA’**). Even simple grammatical features such as the end of the sentence at the start of measure 16 help people in understanding the grouping structure of the music.

Other textual features of the music might not play a large role in the grouping or analogy structures formed by Musicat, but still add greatly to the human listening

experience. For example, the alliterations present in “softer / starlight”, “warmer / winds”, and “June / gentle” not only add to the poetry of the text, but also increase the cohesiveness of the measures and groups that these word-pairs are part of. Similarly, the text-painting in the melody, such as with the lyrics “heaven and earth” (“heaven” is symbolically set to a high note in the melody, while “earth” is on a low note) adds much to the listening experience that is not expressible in terms of the structures Musicat creates.

In any case, bear in mind that Musicat has none of the information conveyed by the text.

Younger than Springtime, 8-Measure Excerpt

5 Youn-ger than Spring-time___ are you, Sof-ter than star - light___ are you,
War-mer than winds of June are the gen - tle lips you gave me.

The image shows a musical score for an 8-measure excerpt from the song "Younger than Springtime". The music is written on two staves in 4/4 time. The first staff contains measures 1 through 4, and the second staff contains measures 5 through 8. The lyrics are written below the notes. The melody is characterized by its simplicity and the use of alliteration in the lyrics. The notes are mostly quarter and eighth notes, with some rests. The lyrics are: "Youn-ger than Spring-time___ are you, Sof-ter than star - light___ are you, War-mer than winds of June are the gen - tle lips you gave me." The underlines in the lyrics correspond to the musical phrases.

Figure 7.2: Younger than Springtime, 8 measures.

I started by running Musicat once on the very first eight measures of the melody:

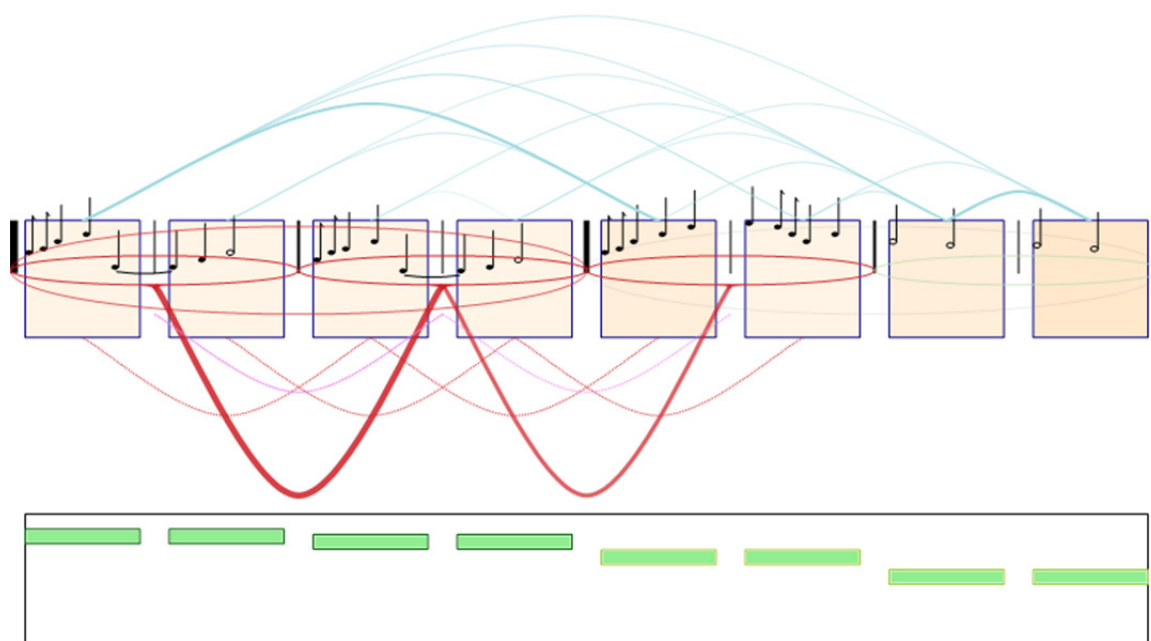


Figure 7.3: Younger than Springtime, 8 measures.

The first four measures have formed a nice, strong pair of groups enclosed by the meta-group (1–4). The next four measures also have the same structure, but group (7–8) and the meta-group (5–8) are very weak and hard to see in the figure. A strong analogy, (1–2)↔(3–4), has formed. This was expected: (3–4) is an exact transposition of (1–2), one step down. A second analogy, (3–4)↔(5–6), is weaker, and this also makes sense: measure 3 starts just like measure 5, but measure 5 continues up where measure 3 (like measure 1) leaped down to a tied note. Measures 5–6 constitute a development of the previous 2-measure pattern, and they drive forward, pushing with a quicker rhythm to reach higher notes. Thus, the weaker analogy makes sense: the previously detected pattern was altered significantly in measure 6, so this analogy should be weaker. Measures 7–8 are not associated with an analogy, which makes sense: they are quite different from anything that came before.

The analogy structure discovered by the program seems to represent a quite cogent hearing of this passage. The grouping in the final four measures, however, is much weaker

than expected. Measures 5–6 seem to set up a momentum (think of Larson’s theory of musical forces) that drives towards the cadence at the end of measure 8. This suggests that we hear (5–8) as a very strong group. Similarly, the $F\sharp^7$ at the end of measure 7 is so unstable that it seems to drive the music forward and should help to solidify group (7–8).

In the next section we see how Musicat does with a longer excerpt made up of the first 16 measures of the melody (the present 8 measures form the first half of the longer excerpt).

Younger than Springtime, 16-measure Excerpt

5 Youn-ger than Spring-time are you, Sof-ter than star - light are you,

9 War-mer than winds of June are the gen - tle lips you gave me.

13 Gay-er than laugh - ter are you, swee-ter than mu - sic are you.

An-gel and lo - ver, hea - ven and Earth are you to me. And when your

Figure 7.4: Younger than Springtime (16-measure excerpt).

The notes in measures 9–15 are identical to those in measures 1–7. Measure 16, however, is different from measure 8: the phrase reaches a conclusion on beat 1 in measure 16, and then the final 3 beats of the phrase are really pickup notes to the next section

⁷ Accidentals such as this “ \sharp ” are not displayed in Musicat’s screenshots, but the program does indeed “hear” them; this is just a simplification in the display.

(although for the runs that follow, the program is given these notes as the final notes of these 16 measures, and it will unfortunately try to include these notes in groups and analogies even though they logically belong to measure 17 as pickup notes). But aside from this small difference, the two halves are very similar. Will Musicat hear this excerpt that way?

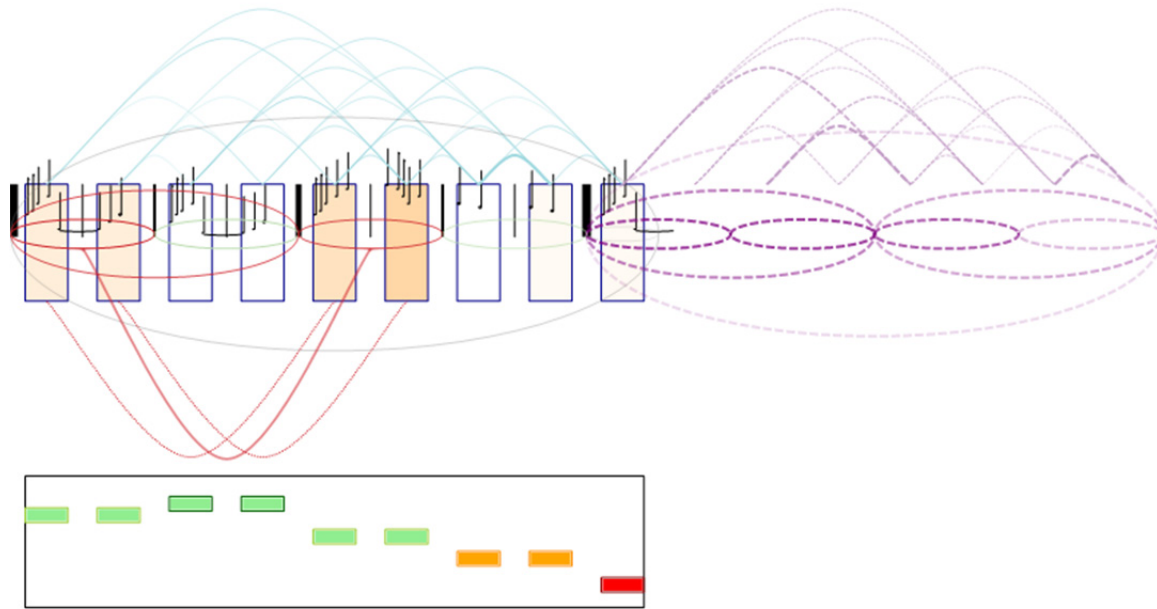


Figure 7.5: Younger than Springtime (16 measures, run 1), after measure 9.

This figure shows the program mid-run, allowing us to see what has happened during the first eight measures before proceeding. Turning to groups first, we see that their overall structure is similar to that of the previous run, although in this case group (5–8), which was weak in the previous run, is so weak here that it cannot be seen in the figure. Group (3–4) is also very weak here, as is (7–8), just as in the previous run. However, in this run an 8-measure meta-group, encompassing most of the melody heard so far, did form (although it is also quite weak). The entire group structure has induced an expectation for these 8 measures to repeat.

Moving our attention to the analogies, we see that only one, $(1-2) \leftrightarrow (5-6)$, has formed so far. This stands in contrast to the previous run in which we saw the nice chain of analogies $(1-2) \leftrightarrow (3-4) \leftrightarrow (5-6)$. At this point in this run, the middle group, $(3-4)$, has been left out of any analogies, probably because its strength was low.

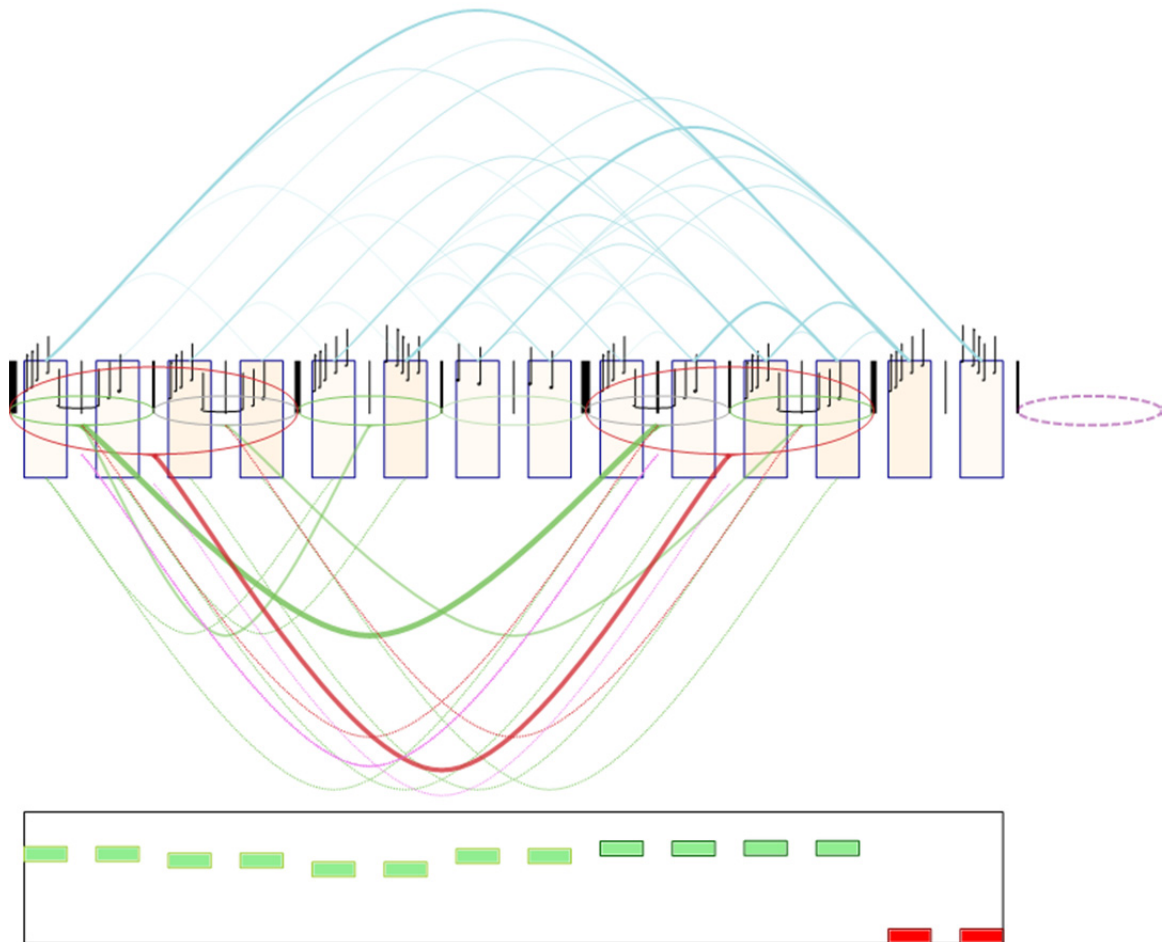


Figure 7.6: *Younger than Springtime* (16 measures, run 1) after measure 15.

Moving forward several measures into the second half of the melody, we see that some long-distance analogies have formed between groups in the first half and groups in the second half. Recall that this second half (measures 9–16) is nearly identical to the first half (measures 1–8), so we might expect a mapping of every structure in the first half to its copy

in the second half. At this point in the run, groups (1–2) and (3–4) in the first half have indeed been mapped onto their corresponding groups, (9–10) and (11–12), in the second half (green analogies in the figure). In addition, a most promising larger-scale (red) analogy has formed, involving the parent groups of those groups: (1–4) \leftrightarrow (9–12). But notice that since group (3–4) was perceived as a weak group, the analogy involving it, (3–4) \leftrightarrow (11–12), is also weak. This is reminiscent of the situation in run 2 of the *Sicilienne*, earlier, in which an early weak group resulted in a weak larger-scale analogy.

At this point in the run, some groups that we might expect are missing: the large group (1–8) from the previous run has not formed, nor have the smaller groups (5–8) or (13–14). With this latter group missing, the expected analogy involving it, (5–6) \leftrightarrow (13–14), cannot form (even though the two groups involved are identical). So the program is still missing what seems quite obvious to a human listener: in the second half it is in the process of listening to what is, so far (up to measure 15), an exact copy of what it heard in the first 8 measures. The program did, however, make that big red analogy, thus showing its recognition of similarity of the first 4 measures of the two sections. Unfortunately, it has not made the leap of extending this idea forward and hearing measures (13–16) as a continuation of this repetition of the first half.

There is some hope for a group and then an analogy involving measures 13–14 to form soon, however. Strong blue measure links have formed involving each of these measures. Measure 14 has a strong link back to measure 6, just as expected. Measure 13, however, has a strong link back to measure 1, not the expected measure 5. The link makes sense in that the measures are indeed similar, but it is also, unfortunately, another clue that *Musicat* has not grasped the concept that this half of the melody is just like the first half.

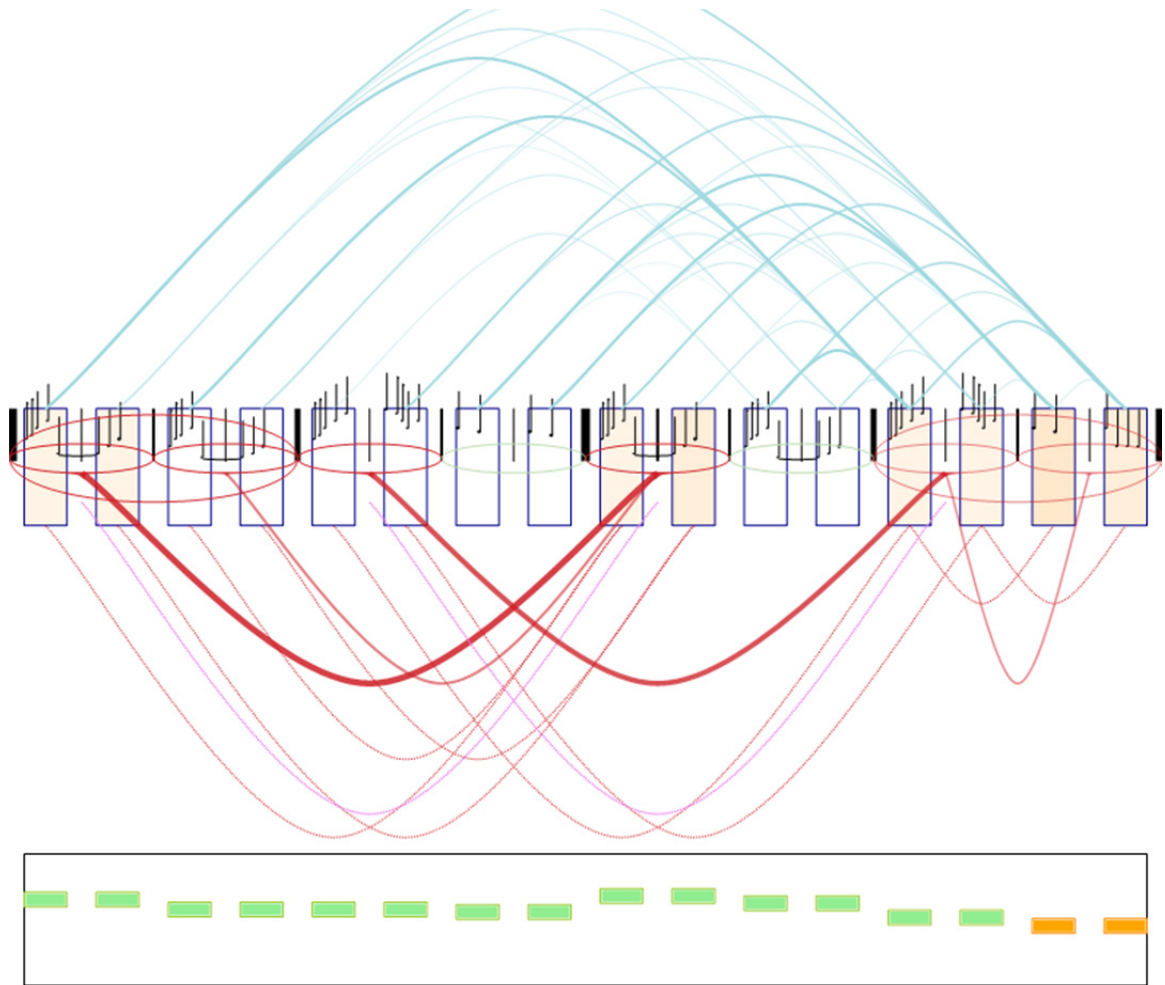


Figure 7.7: *Younger than Springtime* (16 measures, run 1), end of processing.

At the end of the run, in a disappointing development, the promising large analogy $(1-4) \leftrightarrow (9-12)$ has disappeared: because of the weakness of the sub-analogy $(3-4) \leftrightarrow (11-12)$, the larger analogy did not have enough support. Indeed, that smaller analogy was also destroyed. Curiously, the program created another analogy, $(3-4) \leftrightarrow (9-10)$ instead. Notice how now there are two different red analogies connected to group $(9-10)$. This looks silly to a human observer, since *two* different groups, $(1-2)$ and $(3-4)$, are now mapped onto $(9-10)$, while group $(11-12)$ has been left out in the cold. It seems obvious from the picture

that the analogy mapping (3–4) onto (9–10) is extraneous: it looks as if the analogy coming from (3–4) missed its mark on its way to (11–12), landing two measures early on (9–10).

One bit of positive news here is that Musicat did eventually make the analogy (5–6)↔(13–14) that was missing earlier in the run; the link between measures 5 and 13 was discovered, even though measure 13 had previously been linked to measure 1 instead. This analogy, along with (1–2)↔(9–10), is quite strong. If only the analogy (3–4)↔(11–12) and the larger analogy (1–4)↔(9–12) had persisted through the run! In that case we would probably be justified in claiming that the program had noticed the correspondence between the two halves of the melody. But this failure is important because it points out that Musicat can make good medium-size analogies but still remain oblivious to larger-scale patterns. If Musicat were capable of noticing something like “measures 9–11 are the same as measures 1–3... maybe we’re in the middle of a repetition”, then building the rest of the correspondences in the second half of this melody would have been trivial.

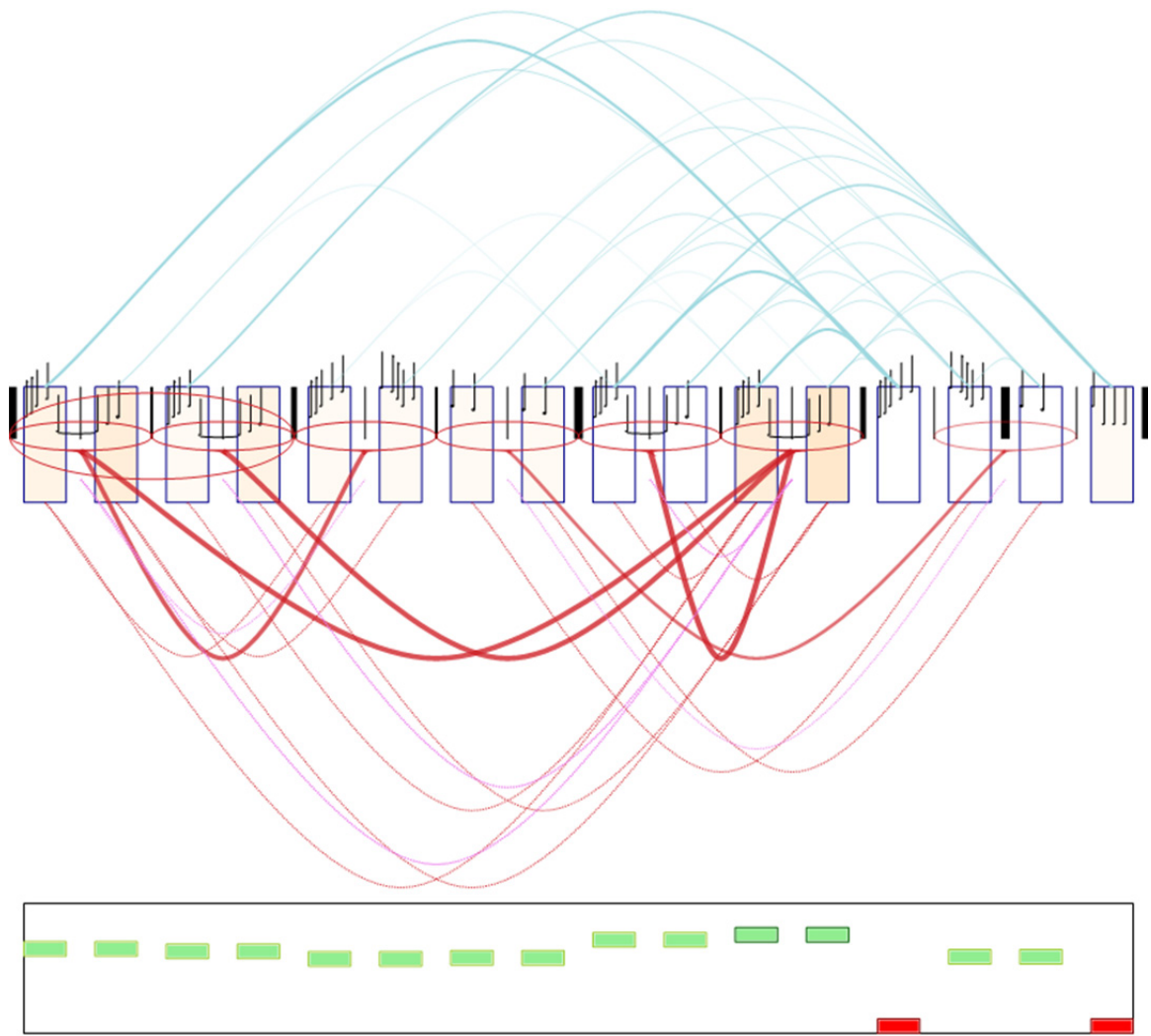


Figure 7.8: *Younger than Springtime* (16 measures, run 2).

I ran Musicat again on the 16-measure excerpt. In this second run, as in the previous run, it misses the essential first-half↔second-half correspondence, and makes another surprising pair of analogies, where two early groups are both mapped onto (11–12). And alas, a very bad group was present at the end of this run as well: not only does (14–15) straddle a very thick bar line, but it also defies the straightforward pattern of grouping every two measures that was established much earlier in the melody. The program has identified some problems in the last 4 measures (see the very low red happiness rectangles associated

with measures 13 and 16), but unfortunately it was unable to resolve its confusion about these measures by the end of the run.

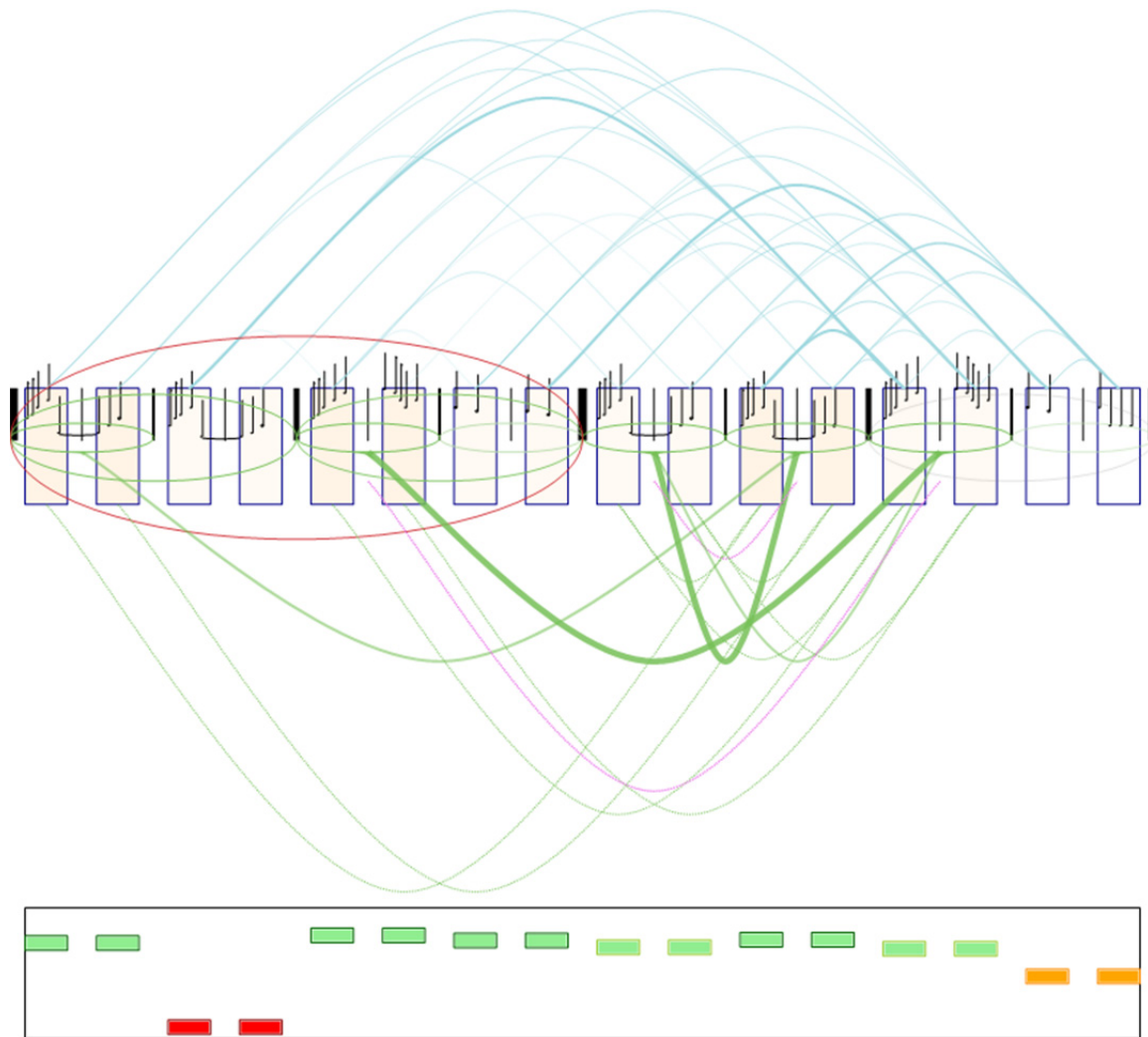


Figure 7.9: Younger than Springtime (16 measures, run 3).

A final run on this 16-measure excerpt exhibits some similar behavior. In this run, the grouping structure was mostly good, but group (3–4) was so weak as to be invisible in the diagram, leading to missing analogies. Some good analogies were formed — indeed, all the analogies present are very reasonable and link identical or obviously-similar groups. The

long-distance analogy (1–2)↔(11–12) is a bit puzzling, however: why didn't the program simply make the mapping (1–2)↔(9–10)? Or (1–2)↔(3–4)? This slightly-off analogy, (1–2)↔(11–12), might serve as a metaphor for the essential problem Musicat is having with this melody: though it is making analogies, it is failing to see the larger-scale structure. The fundamental problem here is the lack of a top-down pressure to simply map measures 1–8 onto 9–16. If this directive were guiding the lower-level processing, it would be simple for the program to make all the smaller analogies such as (1–2)↔(11–12), and it would also help it find the good grouping structure in the second half (providing the first half was heard in a reasonable way).

Younger than Springtime, 8-measure bridge



Figure 7.10: Younger than Springtime, 8-measure bridge.

I ran Musicat separately on the 8-measure “bridge” section that follows the first 16 measures. I have renumbered these measures here (from the original 17–24 to the new 1–8) for readers’ convenience. The program was given measures 1–8 from the figure above, but not the three pickup notes shown in parentheses, since they belong to the earlier 16 measures. Notice that this melody is a case where Musicat would benefit from a more flexible way of grouping that does not limit groups to starting and ending at the same metric position in every measure. The natural grouping structure here would include the 3 pickup

notes before measure 1, and likewise a later group would include the three pickup notes before measure 5. But the final group in this excerpt would need to end at the very end of measure 8 (measure 24 in the context of the entire melody); there are no pickup notes in sight in this measure, and the very next group would start on the downbeat of the original measure 25. The excerpt above, then, should not be exactly 8 measures long, but I was forced to extract exactly 8 measures to give to the program, since it is limited to making groups that are comprised of an integral number of measures. The excerpt given to the program *should* be the $8\frac{3}{4}$ measures shown in the figure, but that is not possible in the current version of the program.

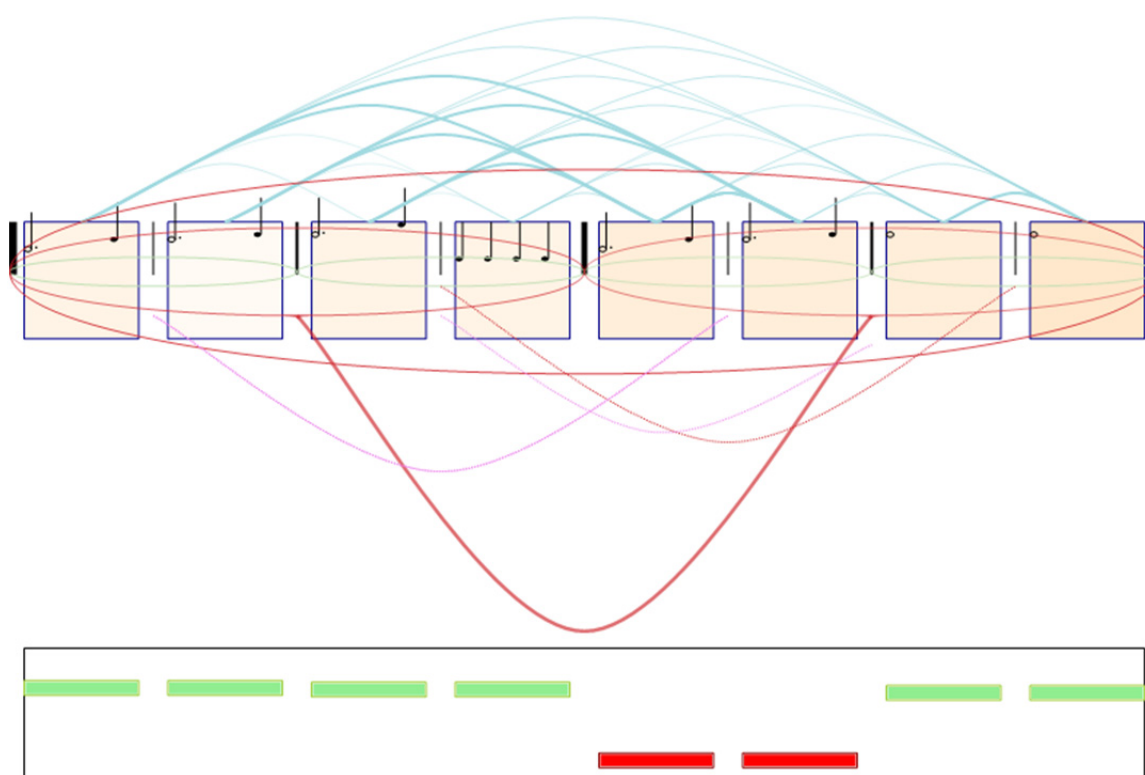


Figure 7.11: Younger than Springtime (middle 8 measures).

Despite the awkwardness of leaving off the initial 3 pickup notes, thereby making the quarter notes in measure 4 seem out of place, the program has created an analogy between the two halves of this melody. Just as the 16-measure excerpt was made of two nearly identical halves, so is this 8-measure excerpt composed of two nearly identical halves. Notice that, unlike in previous examples, the program made a large-scale analogy, $(1-4) \leftrightarrow (5-8)$, without having made any explicit sub-analogies, such as $(1-2) \leftrightarrow (5-6)$, which we might have expected because it would have linked two identical measure-pairs. However, even though “official” sub-analogies were not discovered, the program did indeed find similarities between groups, and, to be sure, finding similarity is indeed analogy-making — I simply use the word “analogy” in a very specific sense when discussing Musicat’s diagrams. One similarity that Musicat saw, for example, was the contour relationship between $(1-2)$ and $(5-6)$ (they have exactly the same notes, and thus exactly the same contour). It also discovered a rhythmic relationship between $(3-4)$ and $(7-8)$, as well as a non-rhythmic relationship of some sort (unfortunately, the type is not indicated in the screenshot), which is surprising because it is between the *two*-measure group $(3-4)$ and the *four*-measure group $(5-8)$. Probably this latter relationship is a tonal relationship indicating the progression from the note D in measure 4 to the dominant, G, in measure 8. In any case, this collection of three relationships, rather than any “official” sub-analogies, was used to support the large analogy $(1-4) \leftrightarrow (5-8)$.

I ran the program again on this excerpt to see if the sub-analogy that I expected, $(1-2) \leftrightarrow (5-6)$, would be created, instead of just the contour relationship that was discovered in this run.

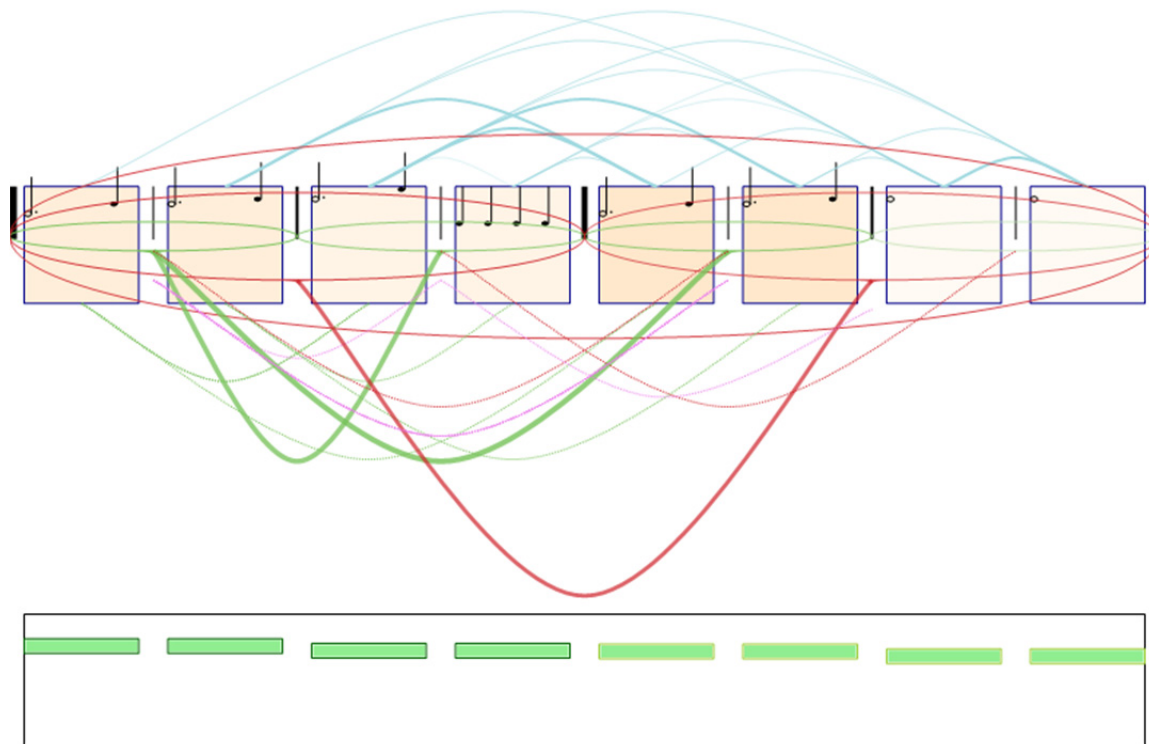


Figure 7.12: Younger than Springtime (bridge, run 2).

Indeed, on a second run the expected analogy, $(1-2) \leftrightarrow (5-6)$, was formed, as was another analogy, $(1-2) \leftrightarrow (5-6)$. Also, the program made the large red analogy $(1-4) \leftrightarrow (5-8)$ again, although this time it has slightly different supporting structures for the mapping. (The *contour relationship* between $(1-2)$ and $(5-6)$ is still present, but it has been augmented with the explicit *analogy* structure $(1-2) \leftrightarrow (5-6)$.) The rhythmic relationship between $(3-4)$ and $(7-8)$ is again present, although the tonal relationship linking $(3-4)$ to $(5-8)$ was not perceived this time.

This listening performance by Musicat seems quite reasonable at first glance, but it did miss one salient feature of the melody: measures 1–3 form a very easy-to-hear *sequence*. Not only is each measure a transposition of the previous measure, one step higher, but the last note of measure 1 is the same as the first note of measure 2, and the same thing happens between measures 2 and 3. Thus, when I hear this passage, I hear measures 1–3 as a

sequence; I certainly don't hear the break between measures 2 and 3 that is implied by the program's grouping. (Incidentally, Hammerstein's lyrics are also incompatible with a group boundary between these measures: the word "invade" straddles the bar line between measures 2 and 3.) In the program's defense, however, to hear this passage as a tonal sequence requires hearing it in the key of G (those F# notes are a big clue, of course, that we have modulated temporarily). Musicat has rudimentary knowledge of tonal functions, but no concept of local modulation, so it would not be able to hear this passage as a tonal sequence unless, perhaps, the G's in measures 2–3 were replaced by G#'s.

ON THE STREET WHERE YOU LIVE (LERNER AND LOEWE)

7 I have of-ten walked down this street be-fore but the pave-ment al-ways

12 stayed be-neath my feet be-fore. All at once am I sev-'ral

stor-ies high, Know-ing I'm on the street where you live.

Figure 7.13: On the Street Where You Live (from the musical *My Fair Lady*).

For our next Complex Melody we consider a short excerpt from the song “On the Street Where You Live”, with music by Frederick Loewe and lyrics by Alan Jay Lerner. This melody greatly influenced the development of Musicat. Not only is it a favorite melody of ours, but also it exhibits many of the features of melody in general that we hoped that the program would eventually be able to “hear”. In its present state, Musicat certainly misses out on many of the interesting details in this melody, but let’s nonetheless see how it fares, even though we know in advance that it will ignore some this melody’s key features. Several

different listening performances follow, to illustrate the various aspects of this melody that Musicat noticed at different times. But before proceeding, I again encourage the reader (as I did in Chapter 3) to stop and listen to or sing through this melody, and to think about what groups and analogies you might be forming during your *own* listening performance.

An important comment about measure numbering is in order: this melody starts on beat 3, so each group formed by Musicat will start on beat 3 of some measure and end just after beat 2 of some later measure. Measure numbering, then, requires extra care: one might think that “measure 1” refers to the two pickup notes at the start of the piece. However, as in earlier sections, measure numbers are indexed with respect to Musicat’s potential grouping points. That is, when I write “measure 1” it will refer to the first 4 beats of the piece, “measure 2” will refer to the next 4 beats, and so forth. I made a metrically-shifted version of the melody (Figure 14) in which the measures exhibit the convention I have just described and the bar lines in this notation correspond to points where Musicat’s group boundaries may occur. Refer to this figure for clarity whenever measures numbers are used in the text.

Figure 7.14: Metrically-shifted version of “On the Street Where You Live”, illustrating the measure-numbering scheme used in the text.

In this melody, there is a small display problem that is apparent in the figures: the ellipses representing groups sometimes appear shifted a bit too far to the left. This problem is a result of an inaccuracy in the graphics-drawing code that shows up for longer melodies with pickup notes⁸, but remember that, despite appearances in some places in these figures, all groups start on the third beat of a measure, continue across a bar line into the next measure rectangle, and continue until the end of beat 2 at a later point in the melody.

⁸ The graphics-drawing code in general was not written to scale well to longer melodies. I had implemented zooming and scrolling capability in an earlier version of Musicat to avoid these problems (see Chapter 8), but the present version is lacking zooming and scrolling, and instead squishes everything horizontally into a fixed-size window.

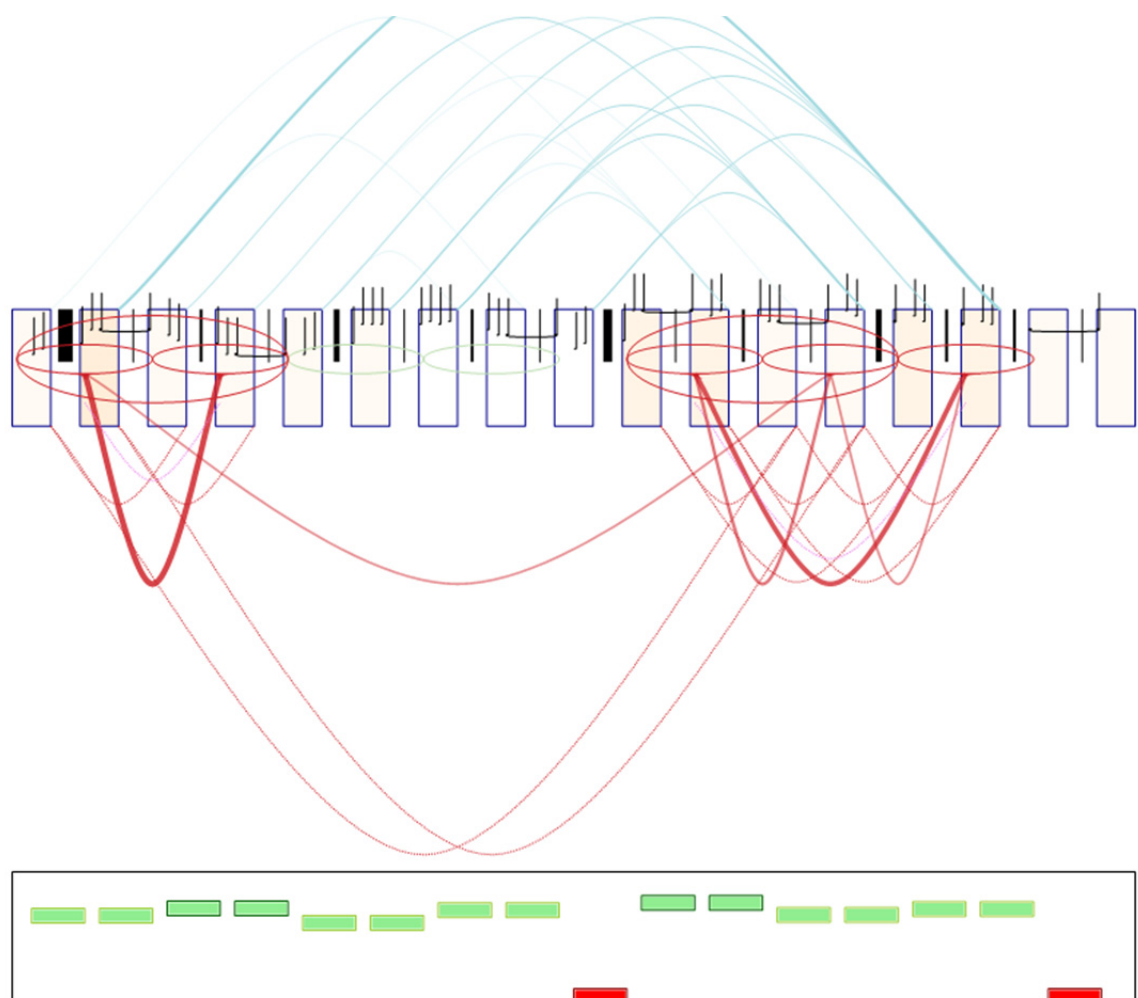


Figure 7.15: On the Street Where You Live (run 1).

The first run had plenty of problems, but it was off to a great start. Measures 1–2 have been grouped together, as have been measures 3–4. A meta-group, (1–4), has formed around these two groups. These two small groups are also involved in the analogy $(1-2) \leftrightarrow (3-4)$. After reading the previous 100 pages or so of this thesis, filled with many examples of analogies made by Musicat, the reader might not be very surprised to see yet another mundane, run-of-the-mill 4-measure analogical structure created by Musicat. However, as simple as this analogy might look, it was surprisingly difficult to get previous

versions of Musicat to make it. The next chapter discusses this in more detail. For now, consider what this analogy means: Musicat has seen the *ascending* melodic line in measures 1–2 (“I have often walked...”) as analogous to the *descending* melodic line in measures 3–4 (“...down this street before”). Specifically, Musicat has noticed that the contour of the first two measures is similar to the contour of the second two measures: they share a pattern of small steps, larger leaps, and note repetitions (step–step–leap–repeat), with the only difference being the direction of motion. Thus measures 1–2 have the contour:

step up, step up, leap up, repeat note

(notes: C–D–E–A–A)

while measures 3–4 have the opposite-direction contour:

step down, step down, leap down, repeat note

(notes: G–F–E–C–C)

Additionally, Musicat has noticed that measure 1 has the same rhythm as measure 3, and likewise that measure 2 has the same rhythm as measure 4. Furthermore, it has noticed that whereas the first group, (1–2), ends on a relatively unstable note (A), the second group, (3–4), ends on a stable note, the tonic (C).

Musicat’s listening performance for measures 1–4 is just what we had hoped for. Measures 5–8, however, pose some problems in this run. First, they have been divided up into two groups, (5–6) and (7–8). This grouping is analogous to that in measures 1–4, and in that sense it is justifiable. However, group (1–2) ends with a long note, which makes the group boundary between measures 2 and 3 quite obvious. Group (5–6), on the other hand, consists entirely of quarter notes, and the notes B–C at the end of measure 6 are repeated at the start of the measure 7, causing a feeling of continuity that links measures 6 and 7

together. During the run, after measure 4, Musicat had an expectation for groups (5–6) and (7–8) to form. I agree that I have this expectation in listening, but between measures 6 and 7 I experience surprise as the tense leading-tone, B incessantly repeats without resolving to a more-stable tone such as C, and then when C finally appears, offering the possibility of tonal closure and a group ending, the melody *immediately* returns to B. All of these details move the melody forward and avoid establishing the expected group boundary between measures 6 and 7. Thus I don't hear a boundary until after measure 8; at that point, I have heard the group (5–8), with no subgroup boundary between measures 6 and 7.

The groups formed after measure 9 in this run are hard to understand. Unfortunately, measure 9 itself didn't end up as part of a group (the red happiness rectangle underneath the measures makes this problem obvious), and then after that point, several groups in a row are shifted over from what we would expect. That is, the groups associated with measures 10–15 would make more sense if they were shifted to the left to span measures 9–14 instead. Just in case the measure numbers are too confusing, I'll restate the problem in terms of the lyrics: the natural grouping I expected to hear in these measures separates the lyrics into these three groups (with groups indicated by parentheses, and meta-groups by larger parentheses):

((All at once am I) – (several stories high)) – (knowing I'm on the street where you live)

The grouping in this run, however, looks like this (including the non-grouped measure 9 at the start):

All at once am ((I several stories) (high, knowing I'm)) (on the street where you live)

Even though the program doesn't have access to the lyrics, this grouping still seems hard to justify when we think of the notes of the melody. Instead of trying to understand the bizarre structures the program made after measure 9, let's move on and look at another run.

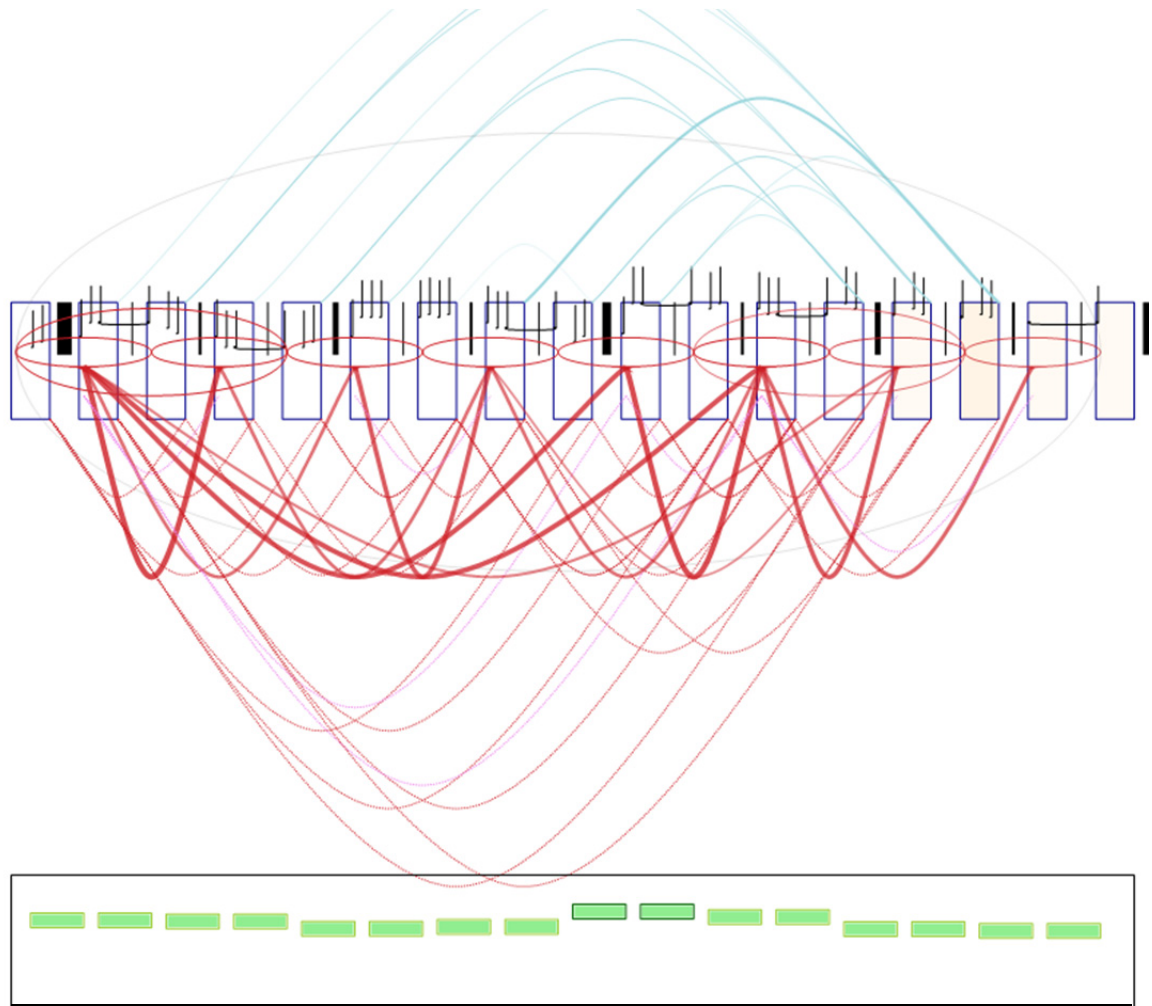


Figure 7.16: *On the Street Where You Live* (run 2).

This run can best be described as “analogies galore!” It looks very different from the previous run, with a large number of groups and a tangle of analogies linking them. The fundamental difference is that here the program has made a group of every successive 2-measure pair: it has formed groups (1–2), (3–4), (5–6), and so forth, all the way through (15–16). (As a frame of reference in these figures, keep in mind that group (1–2) is the first group on the left, and it crosses the thick bar line and continues for another 6 beats after the bar line. Group (9–10) similarly crosses the thick bar line in the center of the melody. And group (15–16), naturally, is the last one in the melody, and hence does not cross the final bar

line.) Two meta-groups have also formed: the expected (1–4) as well as an unexpected group, (11–14). The latter group encompasses the lyrics “several stories high, knowing I’m on the street”, which is a bit strange at the end, although the melody of “several stories high” does indeed sound like it flows nicely into the “knowing I’m” part of the melody, so it is not a completely indefensible group, even though other groupings seem much stronger to me.

Because Musicat heard so many groups in this run, it also had opportunities to make many analogies between the groups. There are so many that the figure above is confusing. I used the detail slider to simplify it:

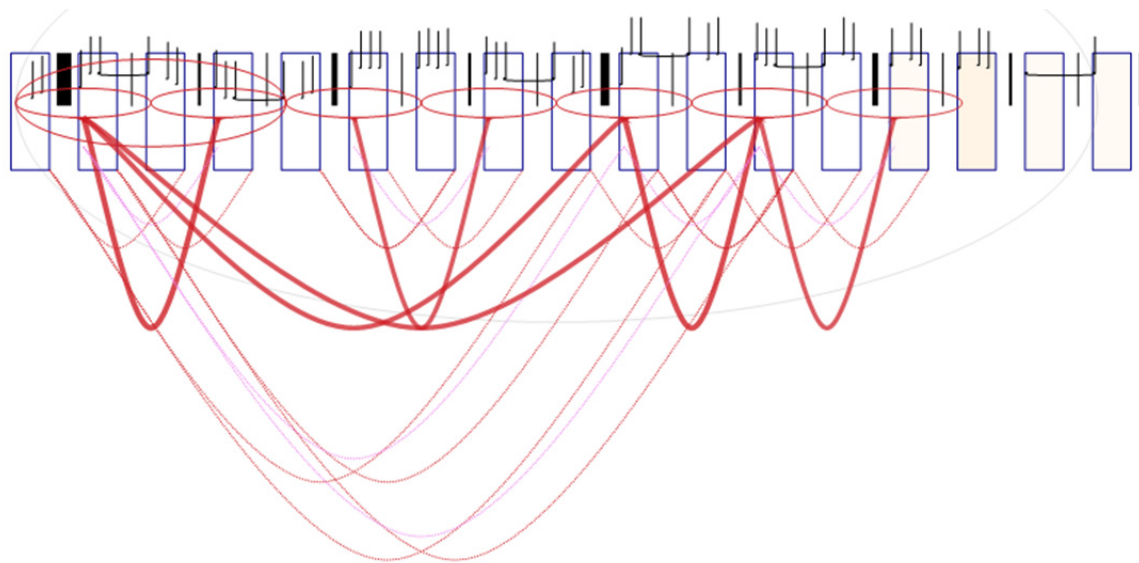


Figure 7.17: *On the Street Where You Live* (run 2), low detail.

This figure shows only the strongest groups and analogies. The group (11–14) is not visible, which is consistent with the idea that this was not a very good group. Additionally, the final group (15–16), has disappeared. This is also not too surprising for me, because a more reasonable group would have been the 4-measure group (13–16) with no 2-measure subgroups. For a human listener, measures 13–15 form a sequence (“Knowing I’m” — “on the street” — “where you live”), and measure 16 involves no note attacks; it has single long

note that has been tied over from measure 15. Therefore, even though it makes sense to hear measures 15–16 as a small group, it makes much more sense for all four measures 13–16 to be grouped together, and ideally heard as a sequence.

In this low-detail figure, there are four analogies that link neighboring groups:

$$(1-2) \leftrightarrow (3-4) \quad (5-6) \leftrightarrow (7-8) \quad (9-10) \leftrightarrow (11-12) \quad (11-12) \leftrightarrow (13-14)$$

There are also two analogies that span long distances:

$$(1-2) \leftrightarrow (9-10) \quad (1-2) \leftrightarrow (11-12)$$

Notice how the first three neighbor-group analogies involve the first three instances of the main melodic theme. I already discussed the first neighboring-groups analogy, $(1-2) \leftrightarrow (3-4)$, because it appeared in the previous run. Musicat found it again this time. The next two analogies Musicat found involving neighboring groups are very similar to this one, because the melody does “the same thing” three times in a row: the first 4-measure group consists of a specific rising and then falling theme, with a characteristic rhythmic pattern. Each of the next two 4-measure segments of the melody is a variant on this initial theme. The analogy $(5-6) \leftrightarrow (7-8)$ is a bit weaker than $(1-2) \leftrightarrow (3-4)$, which makes sense because there is no long note in measure 6 that can be paired with the long note in measure 8. (Indeed, I think that I hear measures 4–8 as a group, and hear an analogy between $(1-4)$ and $(5-8)$ more than I hear the smaller analogy $(5-6) \leftrightarrow (7-8)$.) The next analogy, however, $(9-10) \leftrightarrow (11-12)$, is a strong analogy for Musicat, and it is similar to the strong $(1-2) \leftrightarrow (3-4)$ analogy: there is a long note in measure 10 that pairs with the long note in measure 12, in addition to the similar contour at the start and at the very end of the melody. All in all, then, Musicat’s differing strengths for these three analogies are easy to understand: the first and third analogies are strong, and the second one is weaker. These three analogies,

together, are part of a cogent way of hearing the first 12 measures, and the fact that they remain present even at the low detail level shows that Musicat has “heard” their significance, to some extent. The final neighbor-group analogy, (11–12)↔(13–14), is different from the previous three: whereas the first three analogies each involved the internal structure of a 4-measure segment that was similar to the main theme of the melody, this analogy links the *end* of the third 4-measure segment to the *start* of the final descending sequence in the melody (although Musicat unfortunately did not perceive this final descending sequence). This analogy helps to explain the genesis of what one might think is brand-new musical material in the final sequence (measures 13–16): Musicat’s analogy points out that the start of that sequence sounds somewhat like measures 11–12. This analogy also helped support the 4-measure group (11–14); even though I don’t hear the grouping that way (because it is not as salient to me as the rival group (13–16)), Musicat’s analogy and its grouping of measures 11–14 points out how the last four measures are connected to what came before.

The two strong long-distance analogies in this run both involve the first two measures of the melody, which is unsurprising because those first two measures contain the motif from which the rest of the melody derives. Indeed, the earlier high-detail figure shows many analogies from (1–2) to other groups. The long-distance analogy (1–2)↔(9–10) sounds particularly strong to me as well as to Musicat: groups (1–2) and (9–10) both start just before a thick bar line. Both groups involve an initial stepwise ascent followed by a leap up, and their rhythms are identical. The analogy (1–2)↔(5–6) is also quite important to me, and it was also created by Musicat, but it is not as strong and only shows up in the high-detail picture, because the rhythm of measure 2 is quite different from that of measure 6. The other strong long-distance analogy, (1–2)↔(11–12) is a bit of a surprise, but it is easy enough to understand by thinking of transitivity: there is a strong analogy between (1–2) and (9–10), as well as between (9–10) and (11–12), and this chain of analogies,

(1–2)↔(9–10)↔(11–12), naturally suggests the (1–2)↔(11–12) analogy. However, for a human listener, I think that the (1–2)↔(5–6) analogy is heard much more strongly. Even though people may theoretically understand nearly every pair of measures of this melody in relation to the first two measures, this analogy (1–2)↔(11–12) still strikes me as somewhat unnatural.

In this run, there was no indication, unfortunately, of any relationship *between* the first three neighbor-analogies. (Similarly, the current version of Musicat is not able to notice that the highest note of the first three 4-measure segments is getting progressively higher.) Even though there were many strong 2-measure analogies, the program didn't form anything larger than a 4-measure group, and the large-scale analogies I hoped to see between, say, (1–4) and (9–12) are not present. Will Musicat be able to “hear” these structures on other runs?

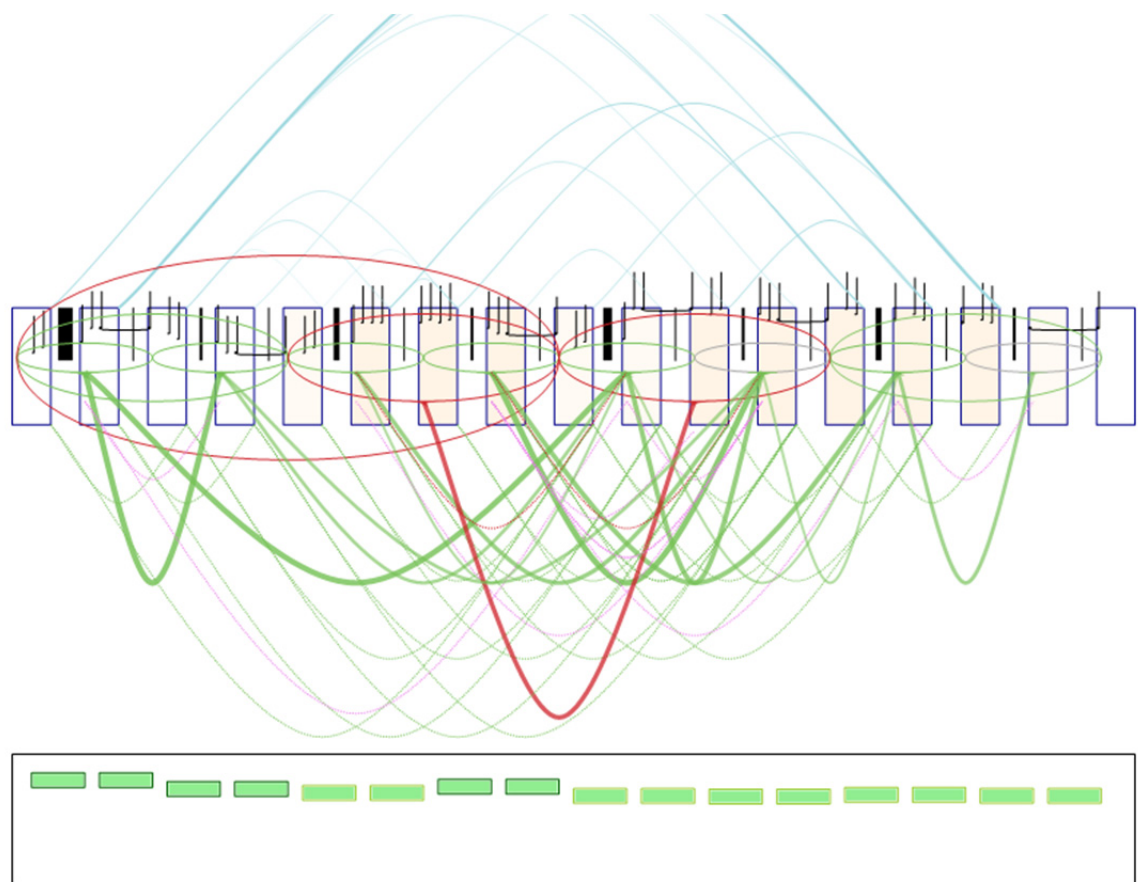


Figure 7.18: On the Street Where You Live (run 3).

In the third run, Musicat did indeed find some larger-scale structures (shown in red). Many of the small analogies found in the previous run were also found here (notice that these small analogies were colored red in the earlier figures, but in this figure we see that Musicat has automatically colored them green to allow the large analogy to be more visible). In this run, however, the grouping structure is very regular: not only are all successive measure pairs grouped together, but also every successive pair of these 2-measure groups is surrounded by a 4-measure meta-group. Furthermore, the first half of the melody has yet another level of grouping: an 8-measure group has formed. It is disappointing that the second half of the melody is not grouped analogously, and that no 16-measure group has

formed, but otherwise the structure makes quite a lot of sense (although as I mentioned previously, I hear group (13–16) without any 2-measure subgroups, and likewise I don't hear a group boundary inside group (5–8).)

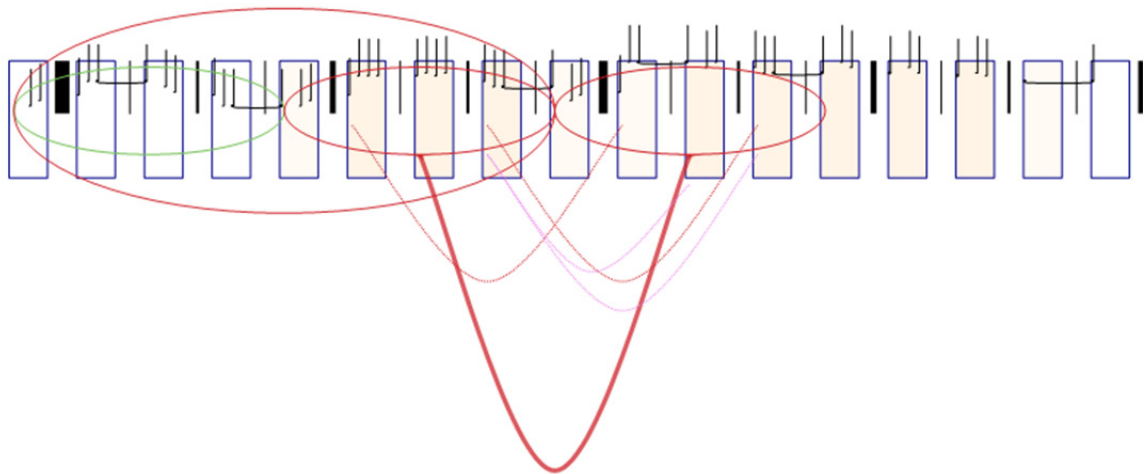


Figure 7.19: *On the Street Where You Live* (run 3), low detail.

I reduced the detail level to highlight the strongest structures in this run. The large analogy (5–8)↔(9–12) and the large groups were evidently the structures perceived by Musicat as being the strongest (except for group (13–16), which doesn't appear in the low-detail figure). The importance and strength of the structures in this picture seem quite reasonable, although it is disappointing that group (1–4) is not involved in any analogies, since it is the first statement of the theme. The strongest analogy that I myself hear is (1–4)↔(9–12), which is missing. Similarly, (1–4)↔(5–8) is quite important but missing. That is, all three of these strong 4-measure groups are closely related, and Musicat has noticed only one of the three possible large relationships. The other important missing structure here is the descending sequence in the final four measures, which Musicat again fails to perceive (in general, the program needs improvement in the sequence domain).

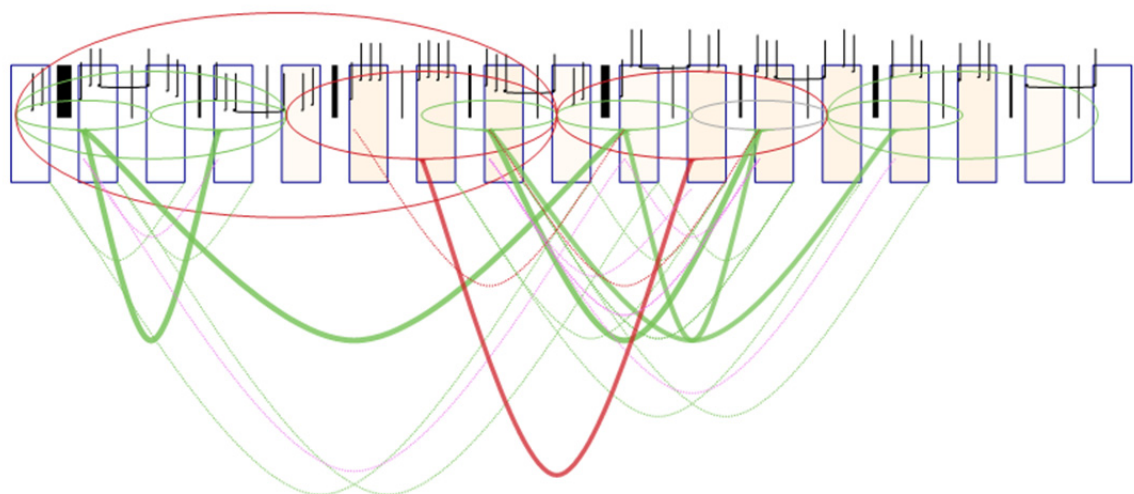


Figure 7.20: On the Street Where You Live (run 3), medium detail.

I raised the detail level to “medium” to see why the large-scale analogies that I expected, involving group (1–4), did not occur. The figure shows that the analogy (1–2)↔(9–10) formed again, as in the last run, but it was only of medium strength, and the program did not find the strong parallel analogy (3–4)↔(11–12); if it had been found, it would have been easy for the program to make the larger analogy (1–4)↔(9–12). Similarly, there are no analogies even of medium strength that link components of (1–4) to (5–8), so no larger-scale analogy had a chance of forming between these groups.

I was happy that the grouping structure of this run was fairly good, and happy about the large red analogy, but I was still hoping that Musicat would find more of these larger-scale analogies, so I ran it again.

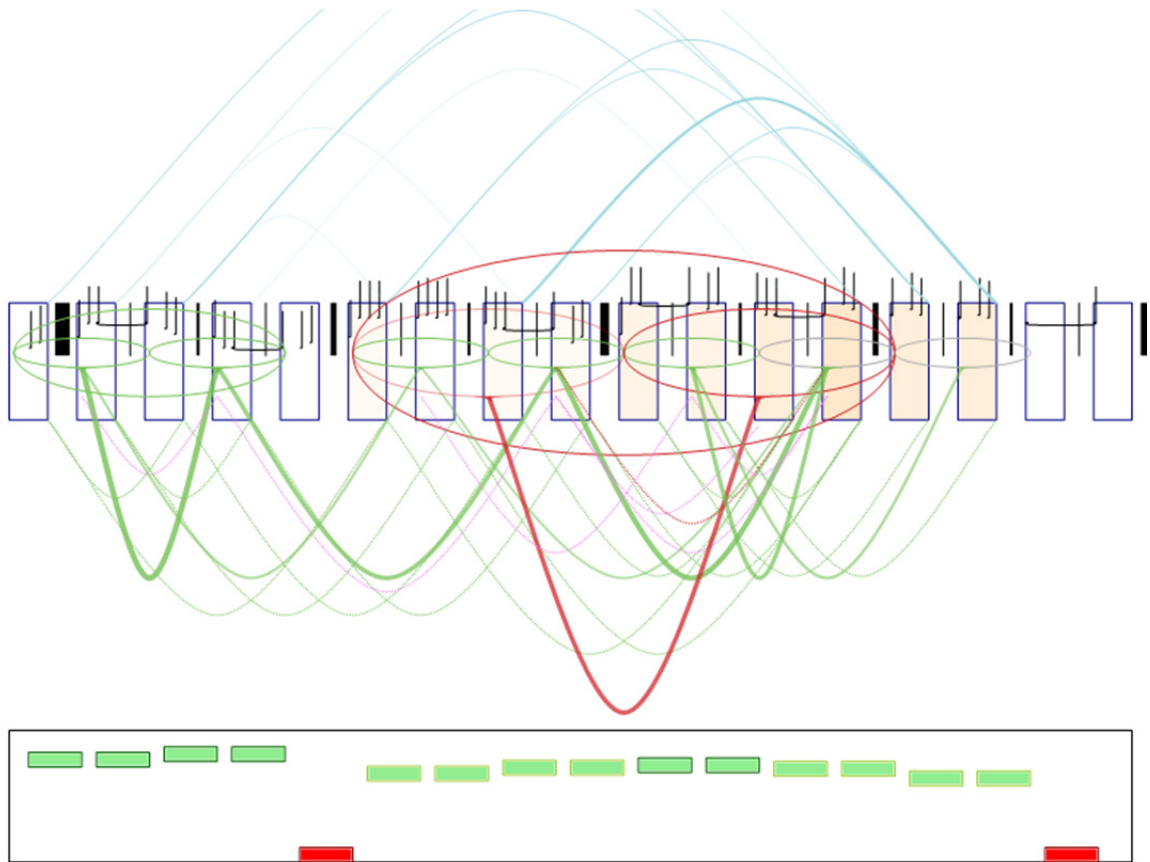


Figure 7.21: On the Street Where You Live (run 4).

At first glance, this run looks similar to the previous one, with the large red analogy in the center. However, in the center of the melody there is a large 8-measure *group* as well, straddling the thick bar line in the center; this is a strange grouping in comparison to the rival groups, (1–8) and (9–16), that might have been formed instead.

Moreover, the red happiness rectangles indicate a serious problem: measures 5 and 16 are not members of any groups. This is a clue to a deeper problem: a closer look at the large red group reveals that it is shifted one measure to the right of the expected grouping — all of the subgroups are out of phase with the regular structure that the program has perceived in the other runs of this melody. For example, the 4-measure red group above is (10–13), not the expected (9–12). This run was disappointing; I ran the program again.

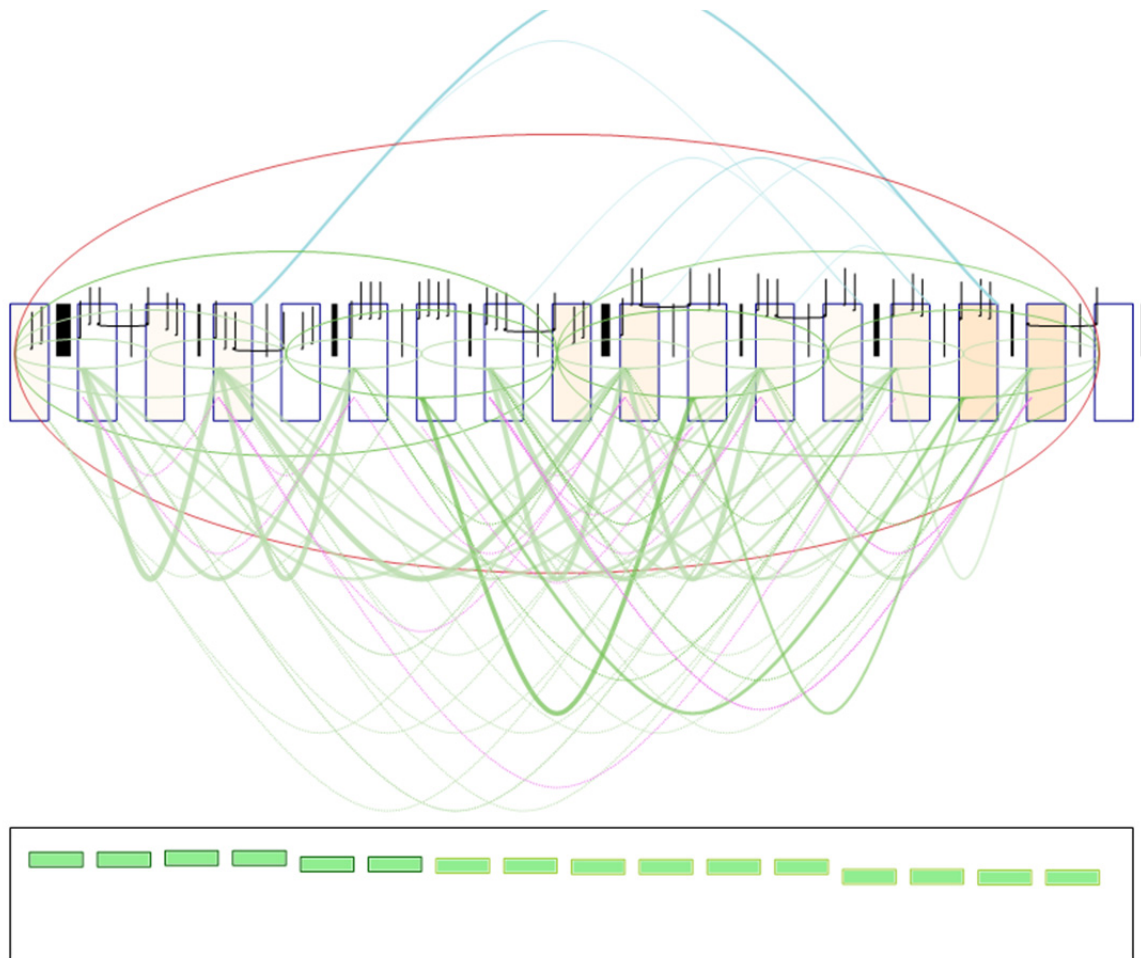


Figure 7.22: On the Street Where You Live (run 5).

This run reminds me of run 3 because of the presence of the large analogy $(5-8) \leftrightarrow (9-12)$, the four 4-measure groups, the large 8-measure group $(1-8)$, and many analogies. In this figure we see even *more* groups and analogies — what a jungle! The missing group $(9-16)$ has finally appeared: Musicat has correctly perceived the melody as composed of two halves, each itself divided in two; there are four 4-measure groups. The entire melody has been grouped together into the very large red group $(1-16)$. The only disagreement I have with this grouping structure is the same thing I mentioned on earlier

runs: I think that (5–8) and (13–16) should not be further subdivided, and instead I hear them as 4-measure groups (and in the case of (13–16), it should be heard as a sequence).

A plethora of analogies was formed in this run, and they seem reasonable, since so much of the melody derives from the opening 2-measure motif. I used the detail slider to focus on the strongest analogies in this tangle of green arcs (here, all the analogies are shown in green because the color red was reserved by the program for the largest structure: the 16-measure group).

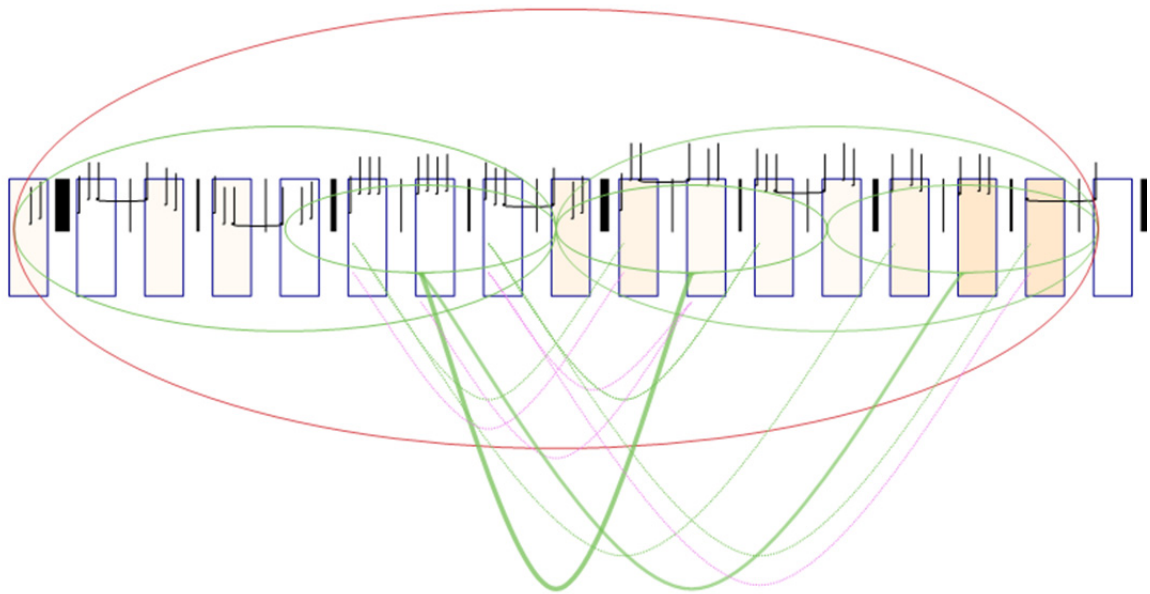


Figure 7.23: *On the Street Where You Live* (run 5), low detail.

It is much easier to see the large analogies in this low-detail view. The analogies (5–8)↔(9–12) and (5–8)↔(13–16) have formed. The first of these is the strongest, as it should be: the analogy involving (13–16) is weaker because (13–16) is rather different from the first three-fourths of the melody. Quite disappointingly, however, the group (1–4) is still not involved in a large analogy. As I mentioned before, it is the first statement of the theme, and the rest of the melody should be heard in relation to it. Indeed, Musicat makes 2-

measure analogies involving (1–2) and (3–4), but misses the larger analogies. In this figure, however, we see that group (1–4) was heard as a weak group in this run, effectively preventing it from being involved in large analogies.

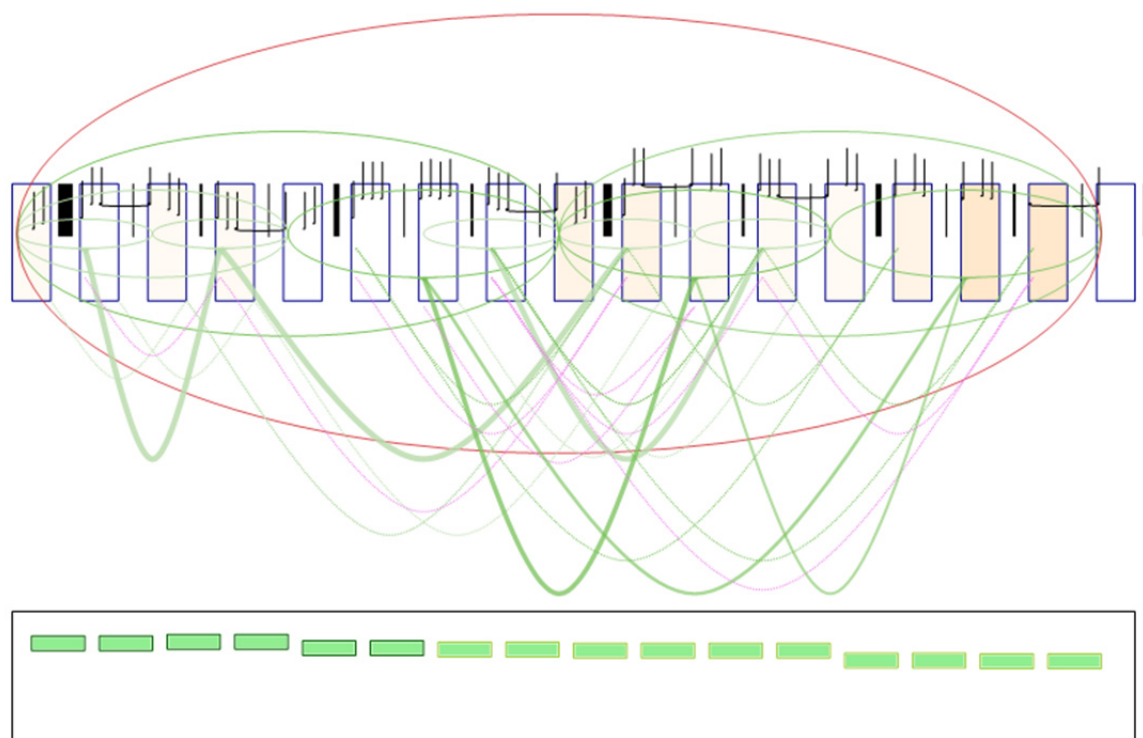


Figure 7.24: On the Street Where You Live (run 5), medium detail.

I raised the detail level to “medium” and saw an additional analogy involving 4-measure groups: (9–12) ↔ (13–16). Thus, all of the possible mappings involving the strong 4-measure groups have been made. That is, Musicat has heard each of these groups in terms of the others. Only the initial theme, (1–4), is missing, but as was explained previously, it is simply because that group was not heard as a strong group, unfortunately.

I decided to run Musicat one more time to see if that first 4-measure group might be involved in a large analogy next time.

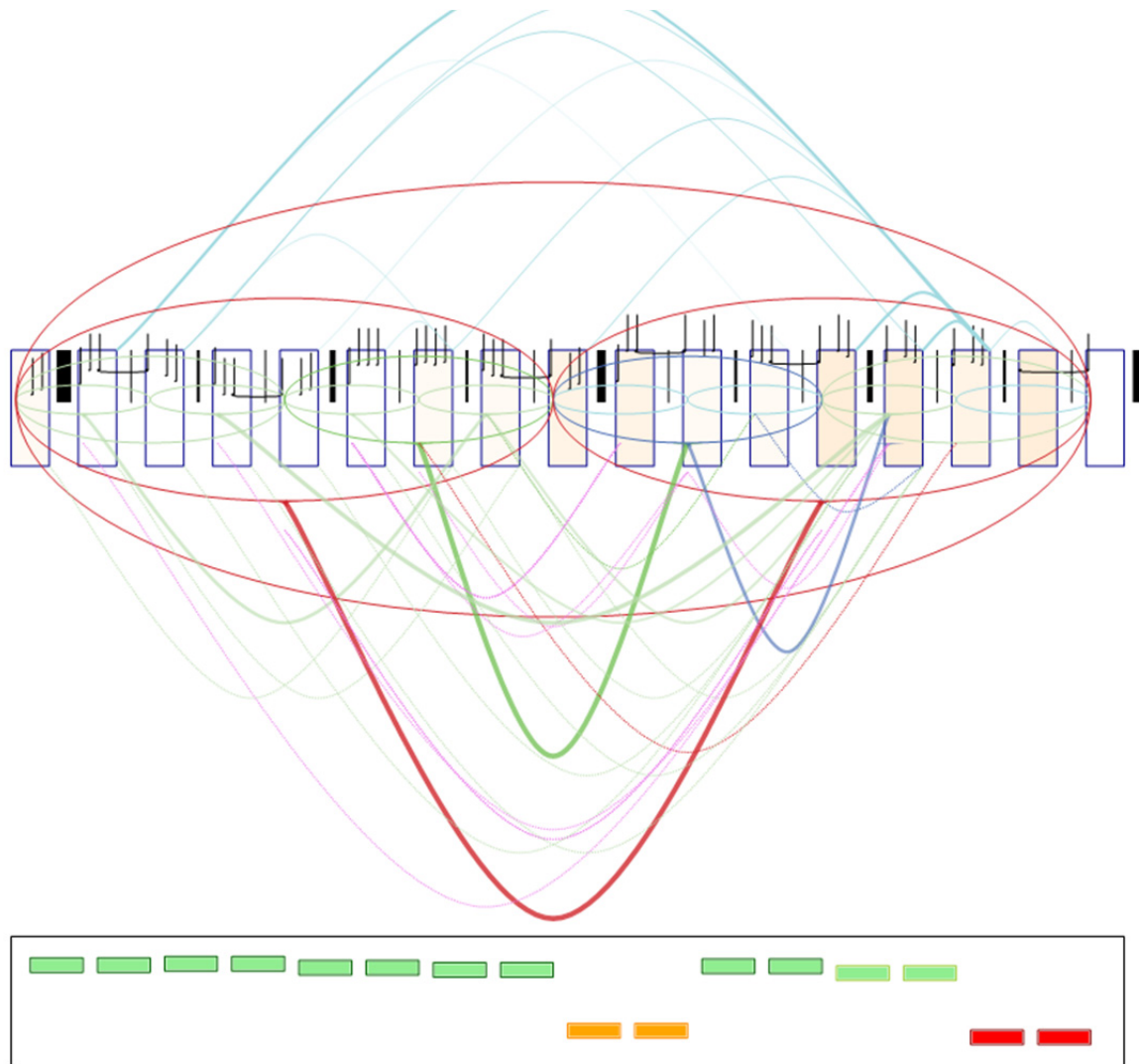


Figure 7.25: *On the Street Where You Live* (run 6), with a very large analogy!

In this run, group (1–4) is again missing the expected analogy. However, in this run that lack seems a bit less important, because a very large analogy has formed! The red analogy here, (1–8)↔(9–16), demonstrates that Musicat has the ability (at least given several listening opportunities) to make rather large-scale analogies. This is the largest structure I have ever seen Musicat create. Back in 2010, it was quite difficult to make Musicat see a small analogy involving just four measures. Therefore, I was extremely pleased in 2012 to see

Musicat create this figure, the likes of which seemed unreachable two years earlier. Just in case it seems trivial for a computer program to make this sort of connection (one might think “Eight measures on the left mapped to eight measures on the right — big deal!”), bear in mind that the program is restricted to a simulation of real-time listening and is attempting to model, albeit in a very coarse manner, some of the constraints of human short-term memory. For instance, on some hypothetical run, Musicat might hear measure 16 and then realize that if only group (1–4) were stronger then the whole red analogy could be strengthened (perhaps by forming the sub-analogy (1–4)↔(9–12)); however, the program is not allowed to go back and retroactively modify groups from the distant past. It is nontrivial for the program to make a 16-measure structure, because the correct substructures have to get put in place more or less in real time, when the relevant measures are first heard or are still fresh in memory. That it succeeded in generating this large analogy was quite gratifying to me.

I do, nonetheless, have two criticisms of this large analogy. First, it should have been stronger: the lack of analogy (1–4)↔(9–12), as in previous runs, is disappointing, and indicates an area for future improvement in the program. Second, although the very large analogy makes some sense, I think a more informed listening performance would avoid mapping the whole first half of the melody onto the second half, and would instead involve hearing the large-scale structure of the melody as three sequence-like instances of the 4-measure theme, followed by a final 4-measure winding-down sequence. That is, I would like the program to hear the melody as having the following formal structure:

$$(A \ A' \ A'' \ B)$$

This form implies that large-scale analogies would be heard between all the *A* segments, and the entire melody would be heard as a group, with no extraneous grouping

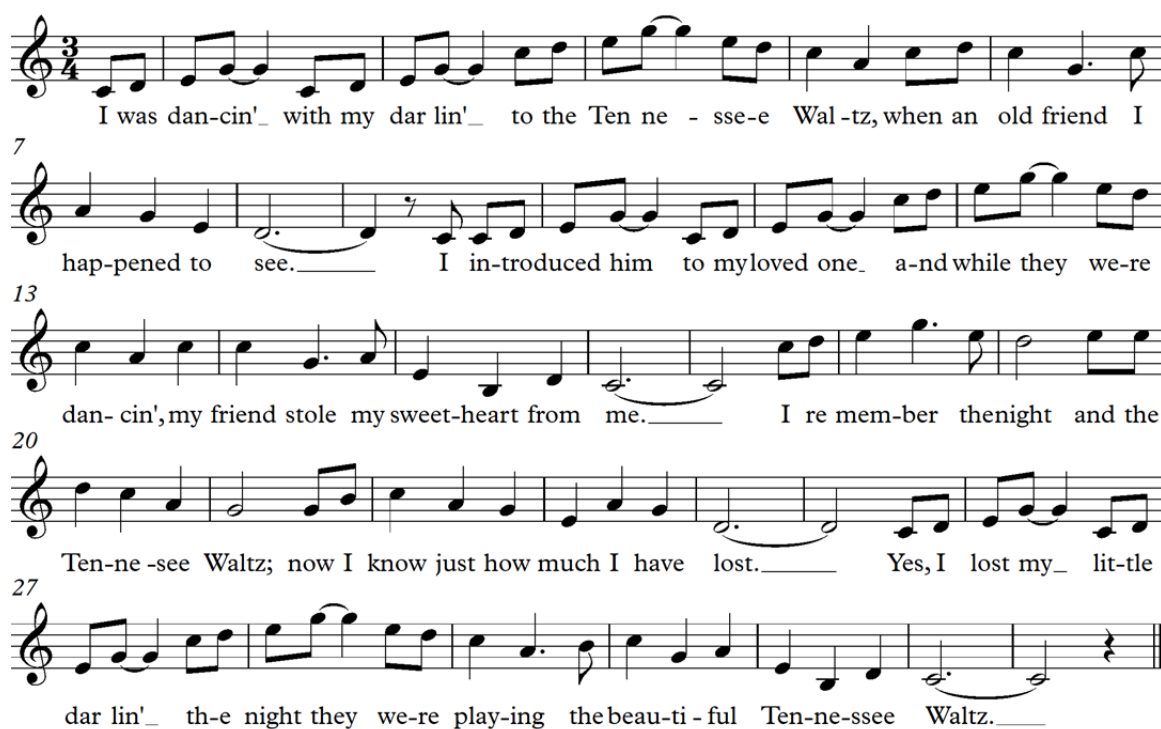
that would break up the progression from **A'** to **A''**. Incidentally, the idea of hearing the entire melody in this way, with three related **A** parts followed by a final **B** part, is very similar to the idea of hearing the sequence in measures 13–16 as composed of three “copies” of a 1-measure group, followed by an extension in measure 16 to complete the phrase. This structural idea of “3 + 1” applies, obviously, to musical structures of various sizes; it is a pleasing coincidence that it appears here in the same melody in two different ways, with structures of two different sizes.

In any case, Musicat’s performance on this melody is encouraging, in that it noticed quite large-sized groups and analogies, especially in comparison with what it saw in the previous chapter’s Simple Melodies. Musicat, however, misses important aspects of this melody, and on some runs fails to make many good structures at all, so it still needs much improvement before it will be able, consistently, to hear the fundamental and salient structures of “On the Street Where You Live”.

The successive runs in this section (starting from run 1 and continuing to this one, run 6) have resulted in pictures of generally increasing size of the groups and analogies formed. This kind of behavior would be expected in a hypothetical version of Musicat that remembered a melody over multiple runs and that was able to build up ever-larger structures with ever greater ease thanks to earlier acts of chunking and memorization of structures. However, the real Musicat forgets everything between runs; in this version of the program, any systematic progression perceived in Musicat’s multiple performances is just an illusion. I also did not include in this section another run of Musicat that had few large groups and that was quite similar a run that had come at the start of the section. If the program had a persistent long-term memory, this kind of behavior — forming larger structures after

repeated listenings — might occur systematically, and Musicat would make richer associations not only across runs on a *single* melody but between *different* melodies.⁹

TENNESSEE WALTZ (STEWART AND KING)



I was dan-cin' with my dar lin' to the Ten ne - sse-e Wal-tz, when an old friend I

7 hap-pened to see. I in-troduced him to my loved one a-nd while they we-re

13 dan- cin', my friend stole my sweet-heart from me. I re mem-ber thenight and the

20 Ten-ne-see Waltz; now I know just how much I have lost. Yes, I lost my lit-tle

27 dar lin' th-e night they we-re play-ing the beau-ti-ful Ten-ne-ssee Waltz.

Figure 7.26: Tennessee Waltz.

Whereas the previous melody, “On the Street Where You Live”, was used throughout the process of developing and testing Musicat, the remainder of the melodies in this section were not considered until the program had reached (or very nearly reached) its present state of development.

In an earlier section, I presented the 32-measure melody “Younger than Springtime” to Musicat in piecemeal fashion, only 8 or 16 measures at a time. Tennessee Waltz also has

⁹ Harry Foundalis’s program Phaeaco, working in the domain of Bongard problems, demonstrates a long-term memory mechanism such as the one suggested here. I plan to implement the idea in a future version of Musicat.

32 measures. I tried letting Musicat listen to the entire 32 measures at once. The results illustrate some of Musicat's problems. The figure was far too tall for the program window, so a large portion of the image is cut off at the top, but this was the least of the program's problems. See the figure below.

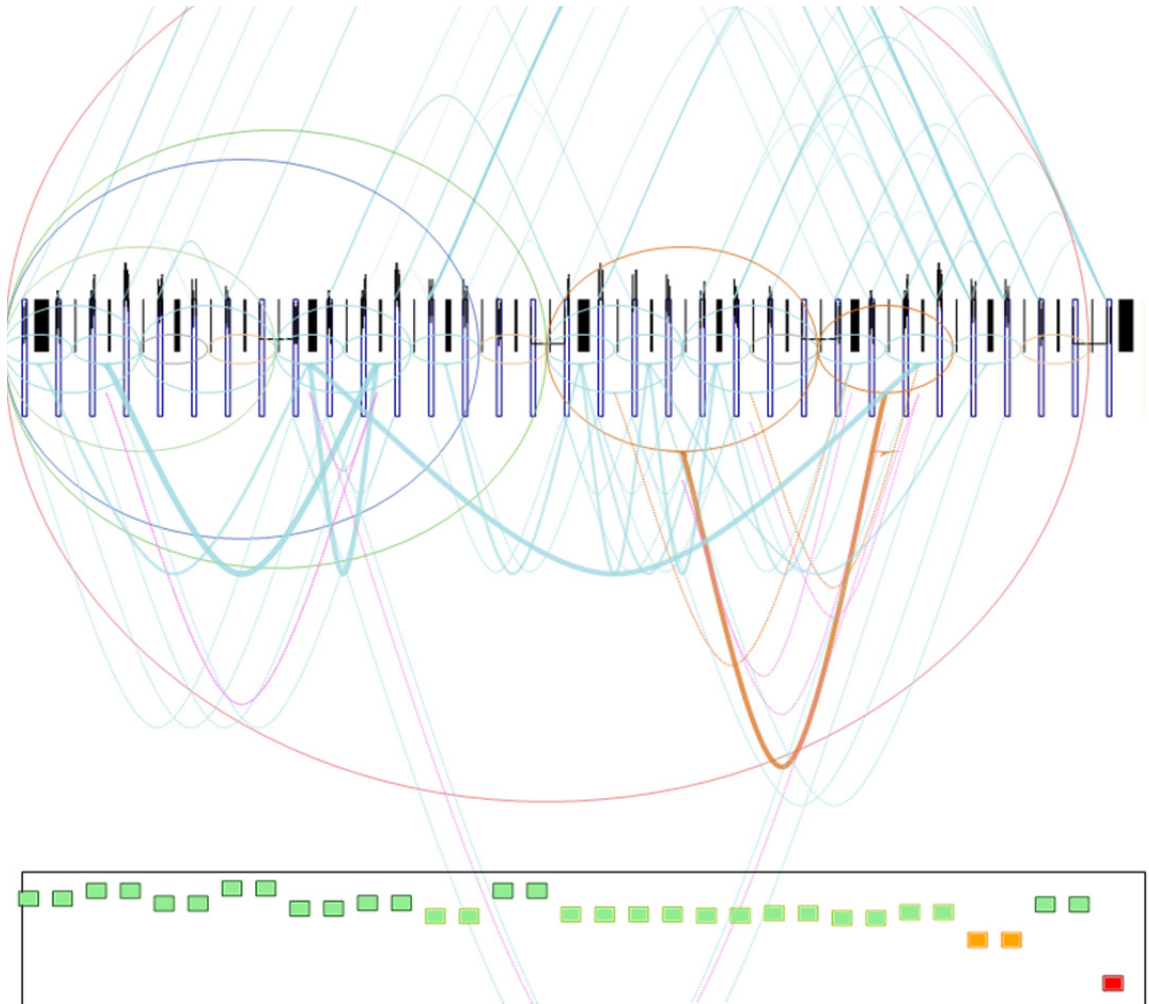


Figure 7.27: Tennessee Waltz.

This strange picture looks more like the mysterious traces of subatomic particles seen in supercollider experiments, or a diagram drawn in some alien language, than it looks like an illustration of musical structure. In part this is due to the lack of support for large

numbers of measures in the current drawing code, but it is also strange-looking because Musicat has created very strange structures. The first 8 measures actually do have a normal-looking structure, and groups such as (1–2), (1–4), and (1–8), are visible, but after this we find the strange-length and quite strong group (1–14) (which is itself enclosed in the much weaker group (1–16), making a very unlikely-looking structure), an orange-colored analogy (17–24)↔(25–28) between groups of quite different lengths (an 8-measure group linked to a 4-measure group), and some long-distance analogies that seem quite arbitrary. Musicat has trouble with melodies of this length, especially if they are complicated. I don’t expect that the problems will forever remain insurmountable, but for the time being I decided to give Musicat shorter chunks than this. Therefore, I split “Tennessee Waltz” into two halves. It turns out, though, that this was still a tricky challenge for the program.

Tennessee Waltz, First Half

I was dan-cin' with my dar lin' to the Ten ne - sse-e Wal -tz, when an old friend I

7 hap-pened to see. I in-tro-duced him to my loved one a - nd

12 while they we-re dan- cin', my friend stole my sweet-heart from me.

Figure 7.28: Tennessee Waltz, first half.

The numbering scheme I use for measures here, as in previous melodies, considers “measure 1” to start on the first note of the melody and to continue for the length of one measure — here, 3 beats. Measure 1 thus contains the notes C–D–E–G (“I was dancin’”), measure 2 contains the next 4 notes, again C–D–E–G (“with my darlin’”), and so on.

Before looking at the program's output, let's consider the structure of the melody for a moment. These 16 measures are made of two nearly-identical parts: measures 9–12 are an exact copy of measures 1–4. Measures 13–16 are slightly different from measures 5–8, but the rhythm pattern is nearly the same, as are most of the pitches. So the big picture expected here, at least by a human listener, is the analogy $(1-8) \leftrightarrow (9-16)$.

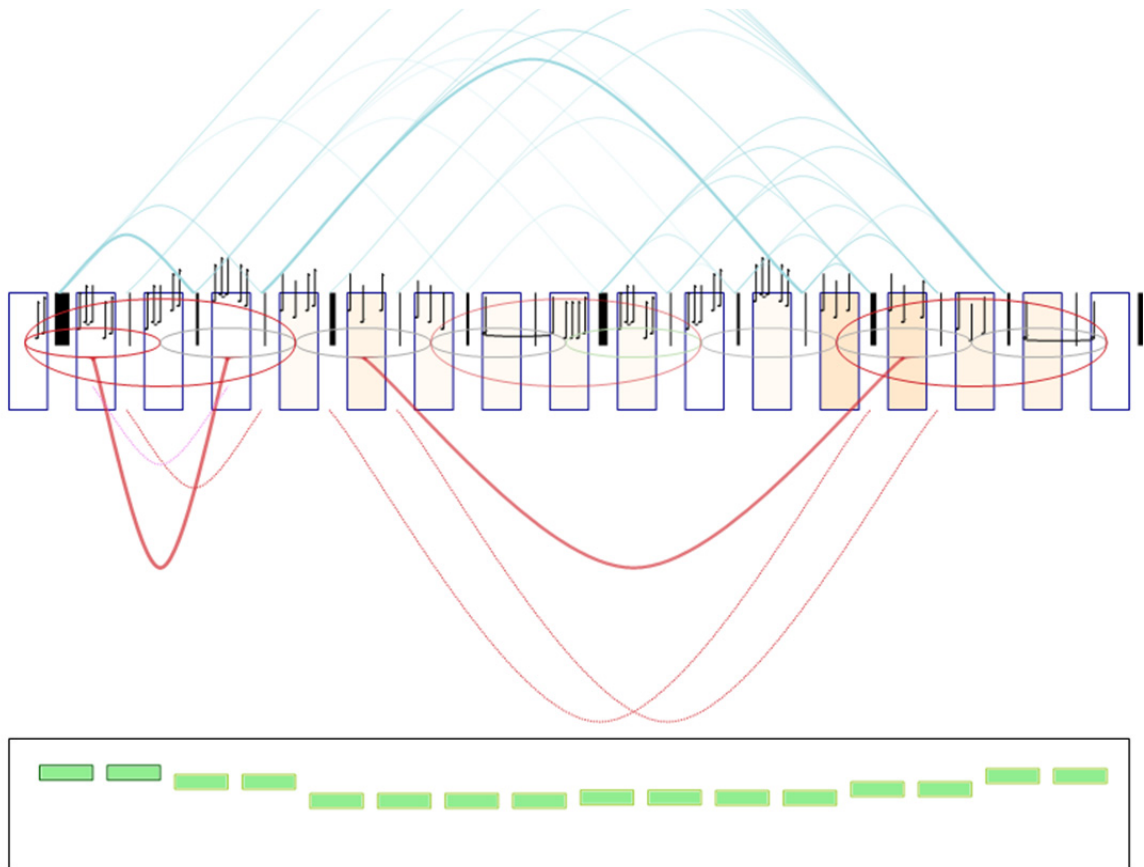


Figure 7.29: Tennessee Waltz, first half.

The large analogy I hoped for is missing in this run. Only two analogies have formed at all — this in striking contrast to all the analogies discovered in “On the Street Where You Live”. I found this puzzling until I remembered all the blue measure links at the top of the picture, which indeed link many of the measures together. That is, Musicat perceived

rhythmic similarities between many pairs of measures, and such noticing of similarities between measures is, as I have pointed out earlier, a limited form of analogy-making. However, for some reason, in this run, the program has not made very many links between structures longer than a single measure. Part of the problem may be the irregular grouping structure it created: the first and last 4-measure groups, (1–4) and (13–16), are the ones I expected. However, a central group, (7–10), has been formed. As happened in some previous runs on other melodies, this group, straddling the center bar line, leads to other strange groups that are hard for the program to match up with other groups in the melody. The two analogies found here, (1–2)↔(3–4) and (5–6)↔(13–14), do indeed link similar melodic structures together, but the obvious analogy — mapping the 8 measures constituting the first half onto the 8 measures constituting the second half — isn't possible when the grouping structure has turned out so strangely, with the single group (7–10) spanning both halves.

I tried a second run on this melody, and got a pleasant surprise (figure on next page).

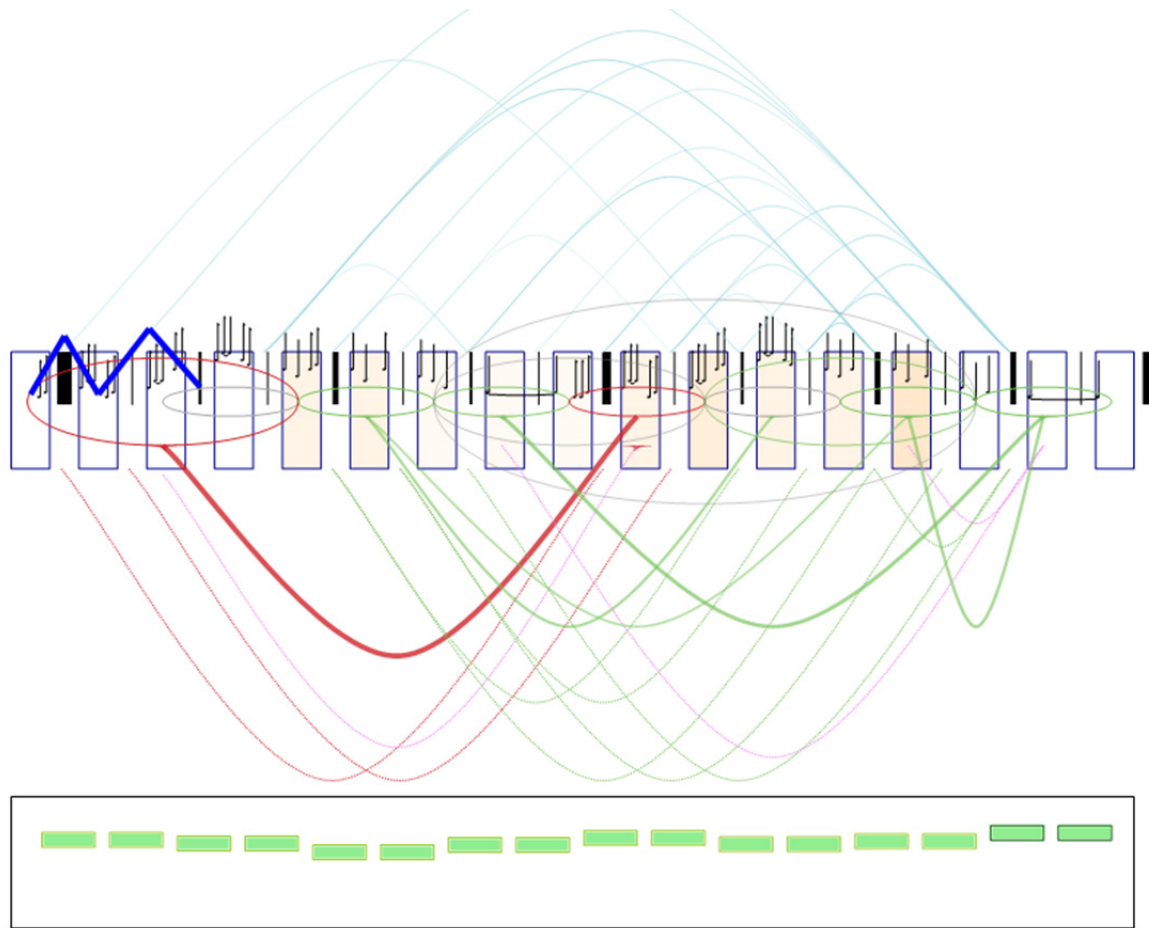


Figure 7.30: Tennessee Waltz, first half (run 2).

In this second run, Musicat discovered a sequence. Not only was this the first time that I saw Musicat find a sequence that was strong enough to persist through an entire run, but I hadn't anticipated or even perceived any sequence myself! A very similar thing had happened during the run of Bad Melody #4 in chapter 5: Musicat discovered a "sequence" that does not involve successive measure transposition (despite the code in the program requiring each successive segment of a sequence to be a transposition of the previous one, by a number of scale steps that is constant for the duration of the sequence.) In Bad Melody #4, each measure in the sequence was identical to the one preceding it, so Musicat perceived a sequence with a 0-step "transposition" between successive measures. In this case, it's very

similar. Measures 1 and 2 are identical, and measure 3 is exactly one octave higher, which Musicat also considered to be identical. Successive chunks in Musicat's sequences must be related by transposition, but Musicat looks only at pitch *classes*, not absolute pitch heights; in other words, octave information is discarded when the program is looking for a sequential pattern. This strategy allows Musicat to recognize as a sequence a passage that jumps between octaves in a circle-of-fifths progression, for instance. In measures 1–4 of “Tennessee Waltz”, then, Musicat saw each measure (or group — the group (1–4) constitutes the third chunk of Musicat's perceived sequence) as a 0-step transposition of the previous, just as in Bad Melody #3; the octave leap between measures 2 and 3 was equivalent to a 0-step “transposition” for the program. Altogether, then, these 0-step “transpositions” allowed Musicat to “hear” a sequence here. It would be easy to fix this behavior if I considered it a bug, but in this case it serendipitously helped the program discover something interesting about the melody. I was quite surprised to see the blue sequence lines in the figure, because I hadn't consciously thought of measure 3 as being the same as measure 1 or measure 2. It sounded, to me, like a much-higher version of the same material, but as a listener I hadn't realized it was simply an exact octave transposition. This surprise was a gratifying instance of Musicat's listening performance informing my *own* understanding of a melody. Thanks, Musicat!

In this run, the program created a few interesting structures, but it missed many obvious things and created some strange structures as well, so the rest of the run was a disappointment. After discovering the surprising “sequence” in measures 1–4, the program *should* have heard measures 9–13 in the exact same way, as a sequence. Instead, it formed two groups, (9–10) and (11–12), instead of linking all four measures together. Worse, these belong to different meta-groups, (7–10) and (11–14). There is another group spanning the thick center bar line of the melody, (7–10), and another one just after it, (11–14), leading to the preposterous group (7–14). This group is truly bizarre, because it includes the cadence of

the first half of the melody, (7–8). Furthermore, Musicat did make the analogy $(7-8) \leftrightarrow (15-16)$, linking the cadences of each half, but it didn't realize that (7–8) and (15–16) were the ends of groups.

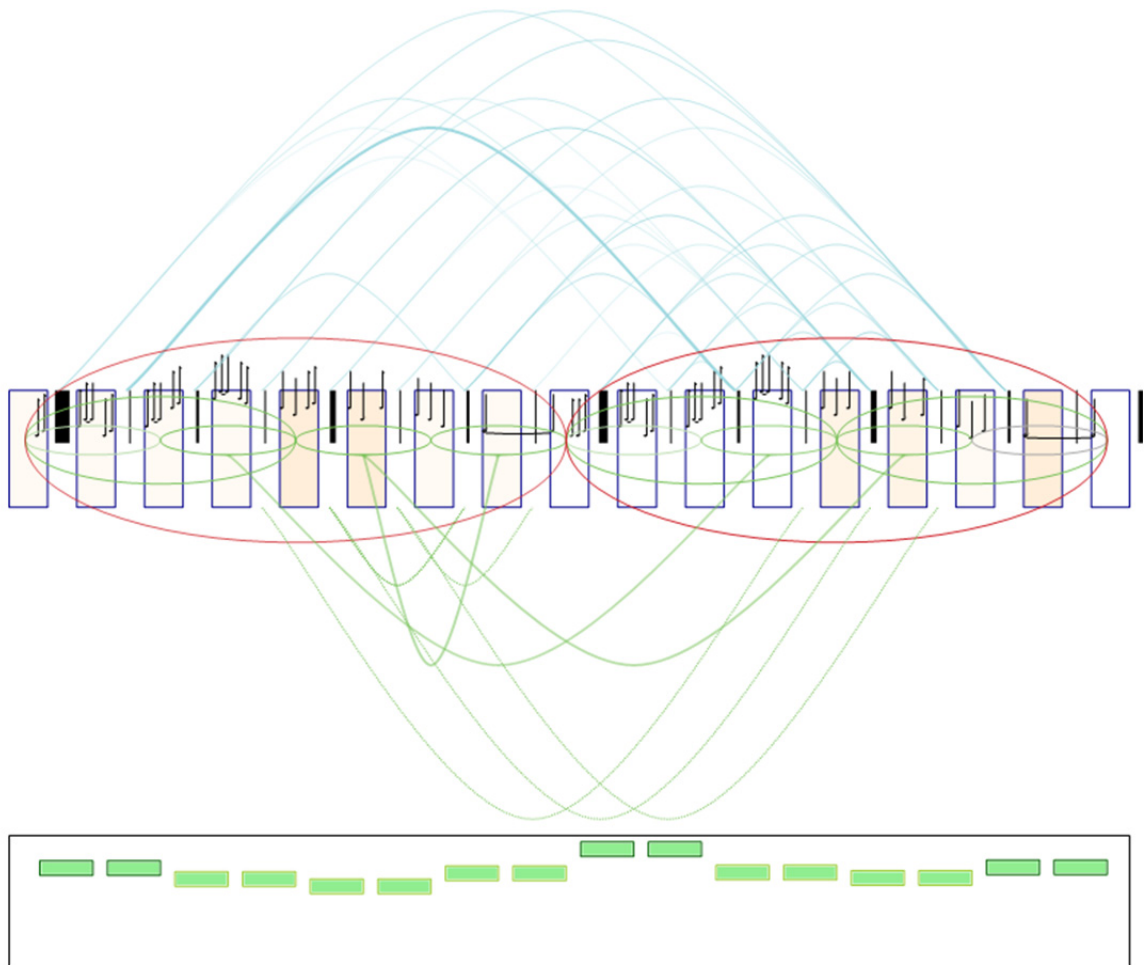


Figure 7.31: Tennessee Waltz, first half (run 3).

On a third run, Musicat again found a sequence (in measures 1–4), but the sequence disappeared early in the run, replaced with two 2-measure groups. But overall, the results were better: the program finally made the hoped-for large-scale grouping structure. The melody has been divided into an 8-measure first half and an 8-measure second half. It also

made two small analogies between the halves: (3–4)↔(11–12) and (5–6)↔(13–14). Unfortunately, though, it missed the most obvious repetition, (3–4)↔(11–12), and the larger-scale analogies (1–4)↔(9–12), and so on. Thus it didn't ever make the *large-scale analogy* (1–8)↔(9–16).

Tennessee Waltz, Second Half

9 I re mem-ber the night and the Ten-ne-see Waltz; now I know just how much I have lost.

13 — Yes, I lost my lit - tle dar - lin' th - e night they we - re

play - ing the beau - ti - ful Ten - ne - ssee Waltz.

Figure 7.32: Tennessee Waltz, second half.

I run Musicat on the second half of the melody (the measures are renumbered here to 1–16 for convenience). Chopping the melody in half this way was frustrating, because one of the important features of this melody, and of many others, is that the final 8 measures here (measures 25–32 in the original non-chopped-up melody) are an exact copy of measures 9–16 in the first half. The program won't be able to see the large-scale **A A' B A'** form of the melody when it hears the two halves separately. Still, however, I was hoping that this run would prove interesting because there are connections to be found between the **B** and **A'** sections.

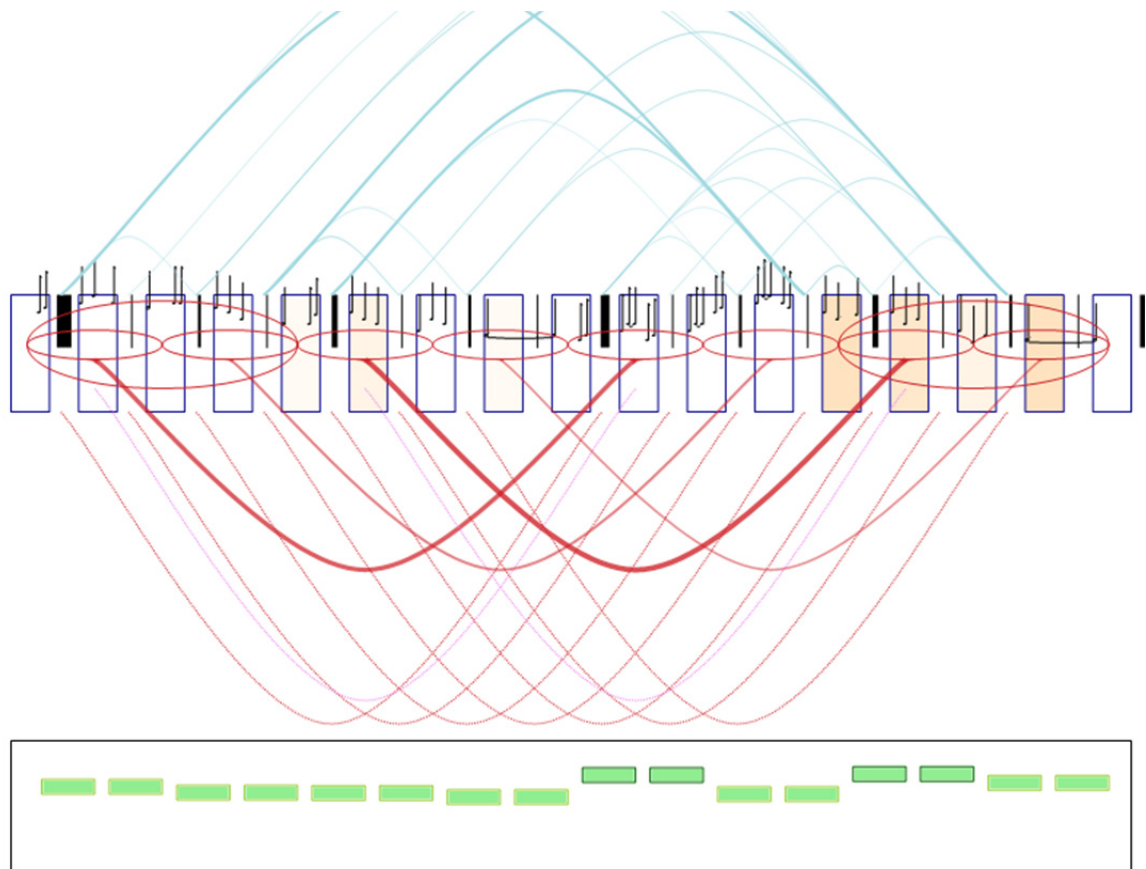


Figure 7.33: Tennessee Waltz, second half (run 1).

The group structure in this run looks reasonable, except that larger groups did not form. The 2-measure and 4-measure groups that did form seem reasonable (although, just as in “On The Street Where You Live”, I myself don’t hear any group boundary between measures 6 and 7, but here the program has formed groups (5–6) and (7–8)). The main theme in measures 1–4 formed a group, as did the winding-down end of the melody, (13–16).

The program found quite a few analogies, although because there were few large groups, all the analogies were between 2-measure structures (measures 5–8 are very similar to 13–16, so that would have been a very likely analogy candidate if the group (5–8) had formed). The first strong analogy in the figure, (1–2)↔(9–10), identifies an important thematic connection between the **B** and **A** material in the melody. Even though measure 2 is quite different from measure 10, measures 1 and 9 are the same melody (with pitches one octave higher in measure 1 than in measure 9), and the program makes an analogy between the two groups because the groups share this starting material. Furthermore, this connection is essentially the same one that led to the sequence in the previous run, measures 1–4. I hoped that Musicat would make the similar analogy between (1–2) and (11–12) in this half because, although these two groups occur at different places in the metric hierarchy, measures 1 and 11 are both in the top octave of the melody and have very similar (but not identical) rhythms and pitch contour. Unfortunately, it didn't find this analogy in this run or the next, but doing so might provide a goal for a future version of the program.

The program found two other easy-to-understand analogies: (5–6)↔(13–14) and (7–8)↔(15–16). (Unfortunately, though, it did not make the obvious meta-analogy (5–8)↔(13–16) here, although the two analogies it did make suggest that it came close.) These connections exemplify something I noticed about “Tennessee Waltz” that made it seems like a really good test for Musicat: the entire melody is made up of just a few basic motifs (exemplified by the rhythms and pitches of the chorus section, measures 1–4 in this half) which are developed slightly to form the rest of the melody (recall the earlier discussion of Arnold Schönberg's book on melody and motif-development). The melody of “Tennessee Waltz”, furthermore, has a casual and imprecise feel to it in spots: different singers (and different versions of the sheet music) use slightly different rhythms and pitches in some sections of the melody (except for critical places like the very start of the song and the chorus

section, which seem more standardized). Listening to two separate and melodically-varying performances of this melody and understanding them as instances of the same song requires — obviously! — a large amount of flexibility in one’s *listening performance*. Hearing an analogies between measures 5–8 and 13–16 (or hearing the two sub-analogies found in this run) requires exactly this sort of flexibility. The following figure compares these two segments of music:

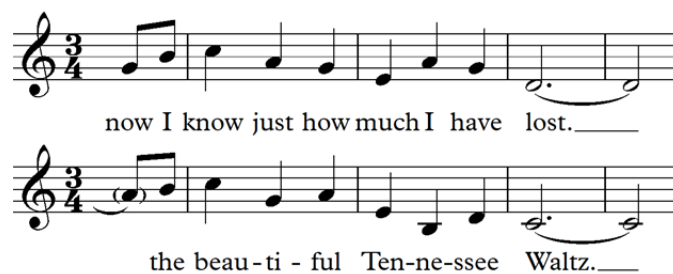


Figure 7.34: Measures 5–8 (top staff) and 13–16 (bottom staff).

These two segments look similar to each other when they are aligned vertically because they both have the same rhythm. This is a nice example of the utility of Musicat’s bias that considers rhythmic somewhat more important than pitch in detecting similarity: even if no pitch information were available at all, these melodies would be heard as “the same”. There are, though, some important pitch similarities: two of the downbeats have the same note in both segments: the C for lyrics “know” and “beau(tiful)” and the E for “much” and “Ten(nessee)”. Most of the less important notes — those on the non-downbeats — are different in the two segments. Notice, for example, the curious swapping of notes A and G in the second full bar of the figure: notes A and G (“just how”) in the top segment become G and A (“-ti-ful”) in the bottom segment. The final downbeat in each segment (with lyrics “lost” and “Waltz”) has the note D, suggesting a dominant chord, in the first segment, and

the tonic C at the end, helping human listeners, as well as Musicat (potentially), to hear these two segments as parts of an antecedent→consequent relationship.

Even though Musicat didn't see the entire analogy (5–8)↔(13–16) in this run, the two smaller analogies it created for these measures show that it is flexible enough to hear similarities despite these melodic differences.

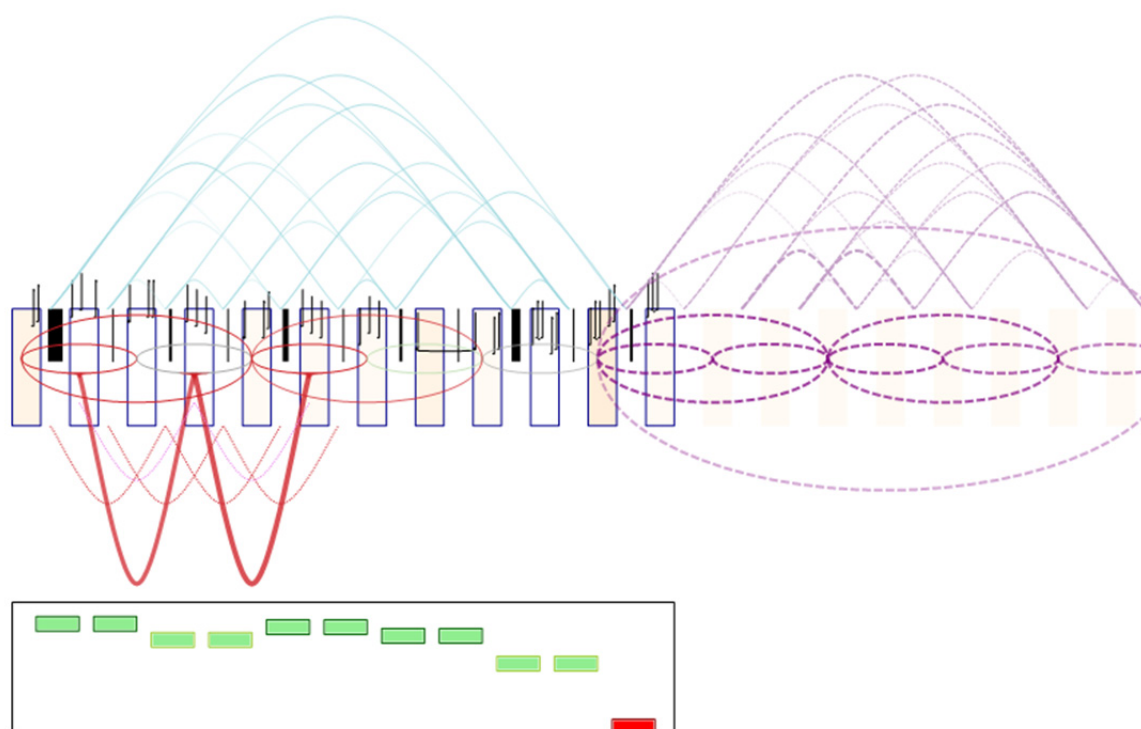


Figure 7.35: Tennessee Waltz, second half (run 2), after measure 12.

I ran Musicat on the second half of “Tennessee Waltz” one more time. Although in this chapter I have been focusing on the *final* results of each run, I think it will give some insight to the reader to see a few measure-by-measure screenshots of this run in progress. With simpler melodies such as “Twinkle, Twinkle” we have already seen how groups and analogies rapidly come in and out of existence as the program listens, but it is interesting to see this happening for larger structures.

During the second run, after measure 12 the groups and analogies of the first 8 measures look good — the two red analogies here were not created in the first run, and the group (5–8), which was missing in that run, preventing the desired analogy, has been formed. The only strange thing is the large purple expectation, which is essentially a copy of the structure in the first 10 measures, which is predicted to occur starting on measure 11. This is unreasonable, because the important group here is the 8-measure group (1–8), not (1–10). Indeed, the strangely sized group (1–10) does not exist in the figure. It must have come into existence briefly — just long enough for the expectation to be generated — only to be destroyed soon after.

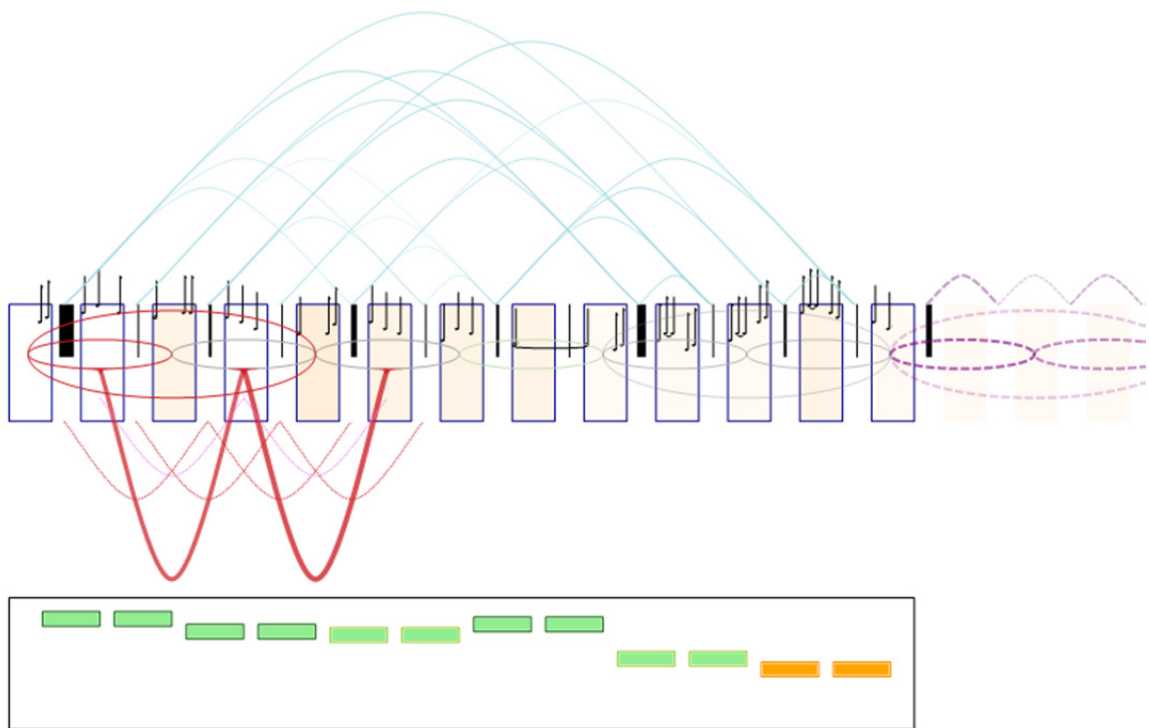


Figure 7.36: Tennessee Waltz, second half (run 2), after measure 13.

Fortunately, a measure later, the large purple expectation has vanished, and has been replaced by a more reasonable expectation for the next 4 measures (which will turn out to be

the final measures of the melody). The 4-measure group (5–8) has, unfortunately, been destroyed, although another one, (9–3), has appeared, but this one is very weak at this point.

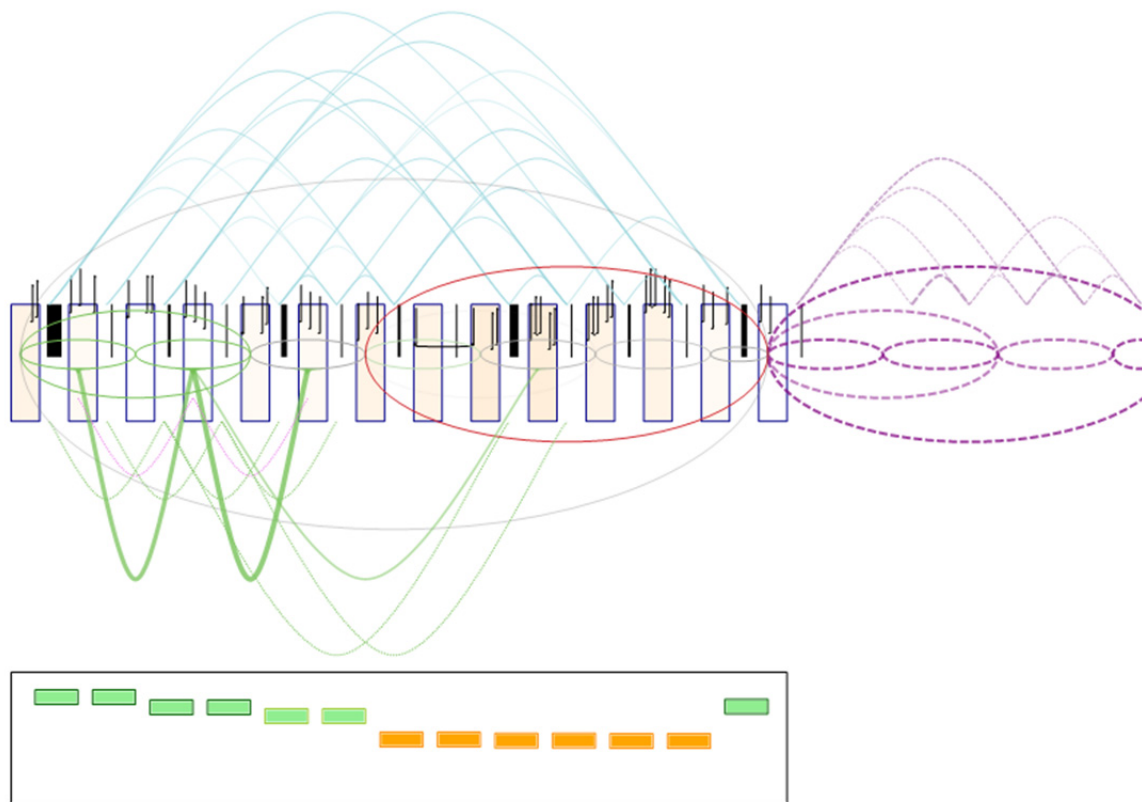


Figure 7.37: Tennessee Waltz, second half (run 2), after measure 14.

After another measure has passed, the group (9–13) has also vanished as has the 4-measure purple expectation. Although these are frustrating developments, they are somewhat understandable as consequences of the dynamic nature of Musicat's perception. A more serious problem is that we have three new large structures, of which two make little sense. First, the red group (7–13) starts on measure 7 for no good reason, and is 7 measures long! Fortunately, the group is weak, as is indicated by the orange happiness rectangles underneath it. This red group has induced a ridiculous purple expectation for another 7-measure

structure starting on measure 14; the previous 4-measure expectation was much more reasonable (and would have eventually helped to form the strong group (13–16), had the expectation not been destroyed and replaced by this one). Finally, a large group (1–14) has been formed. This group is not so strange, as long as in the future it is expanded to include the next 2 measures of the melody, as they are heard.

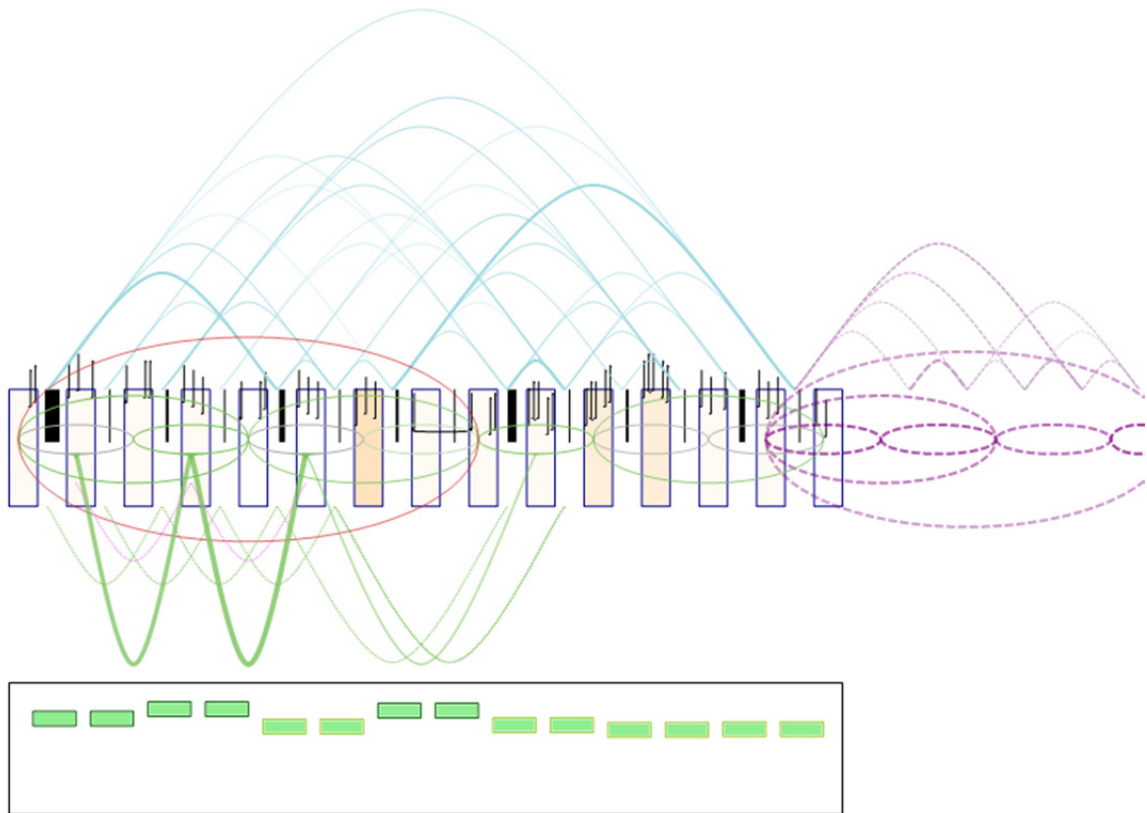


Figure 7.38: Tennessee Waltz, second half (run 2), after measure 15.

After one more measure the weird 7-measure group (7–13) has fortunately been destroyed, and a rival red group (1–8) has formed. I call it a “rival” group because measures 1–8 overlap with measures 7–13, and it is likely that while the program listened to measure 15, these two groups actually had a competition in which group (1–8) emerged victorious.

The strange 7-measure purple expectation is still in place, but notice that Musicat is, so far, sticking with a newly-perceived group (11–14), which overlaps with the expectation. I would have preferred measures 9–12 to have been heard as a *sequence*, as in run 2 of the first half of the whole “Tennessee Waltz” melody, but this group is not unreasonable.

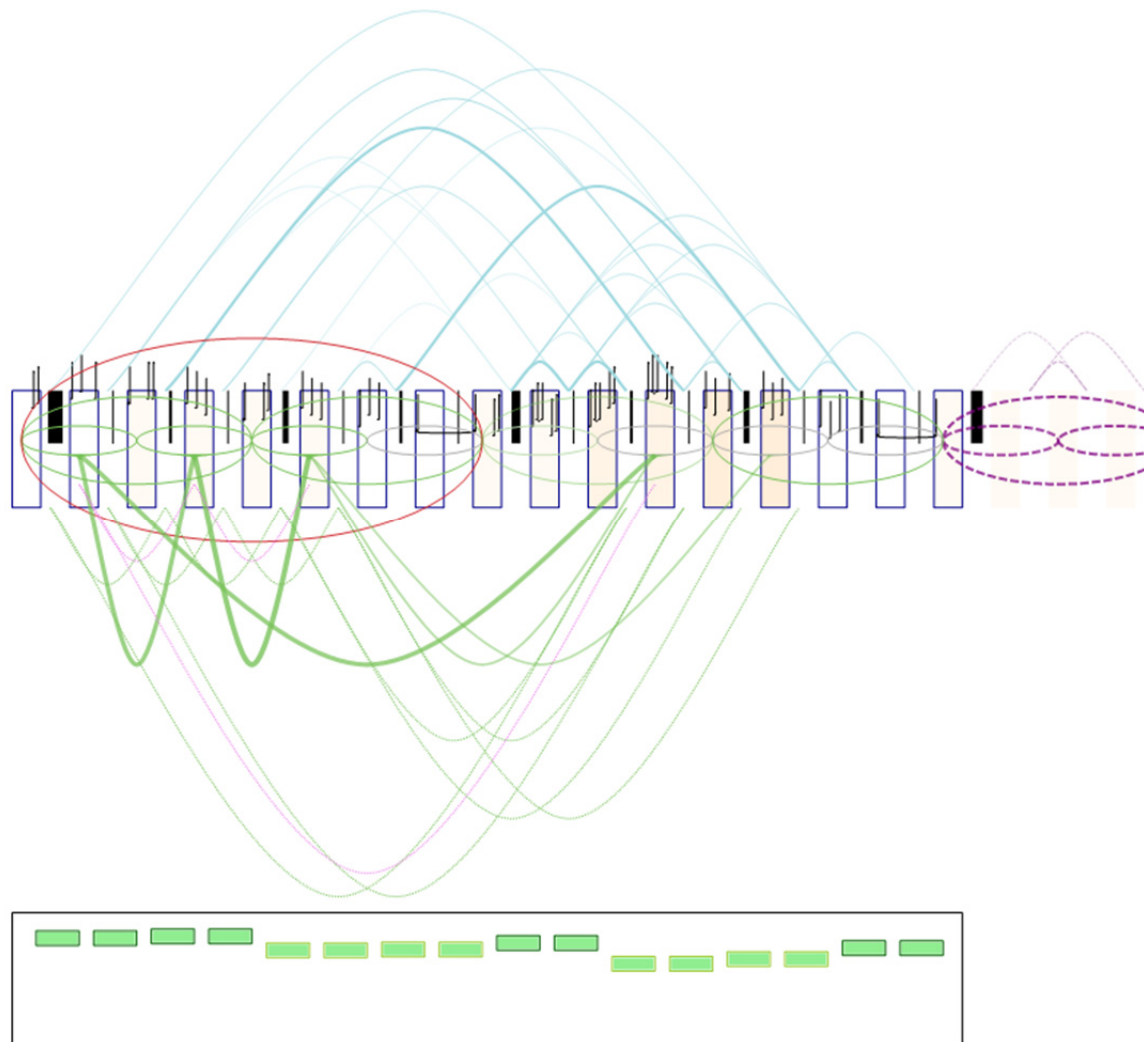


Figure 7.39: Tennessee Waltz, second half (run 2), after measure 16(+3).

After the final measure was heard, Musicat kept running as usual for a few extra measures' worth of time. The figure above shows its state after the third extra measure. The

strange 7-measure expectation has finally been replaced with a 4-measure expectation, which is simply an expectation for the group structure of (13–16) to occur again. Notice that (13–16) is a new group, as is the weak group (9–12) — thus, a regular, repeated group structure has emerged for the entire melody. Unfortunately, the meta-group (9–16), which I expected, didn't form around these two new groups.

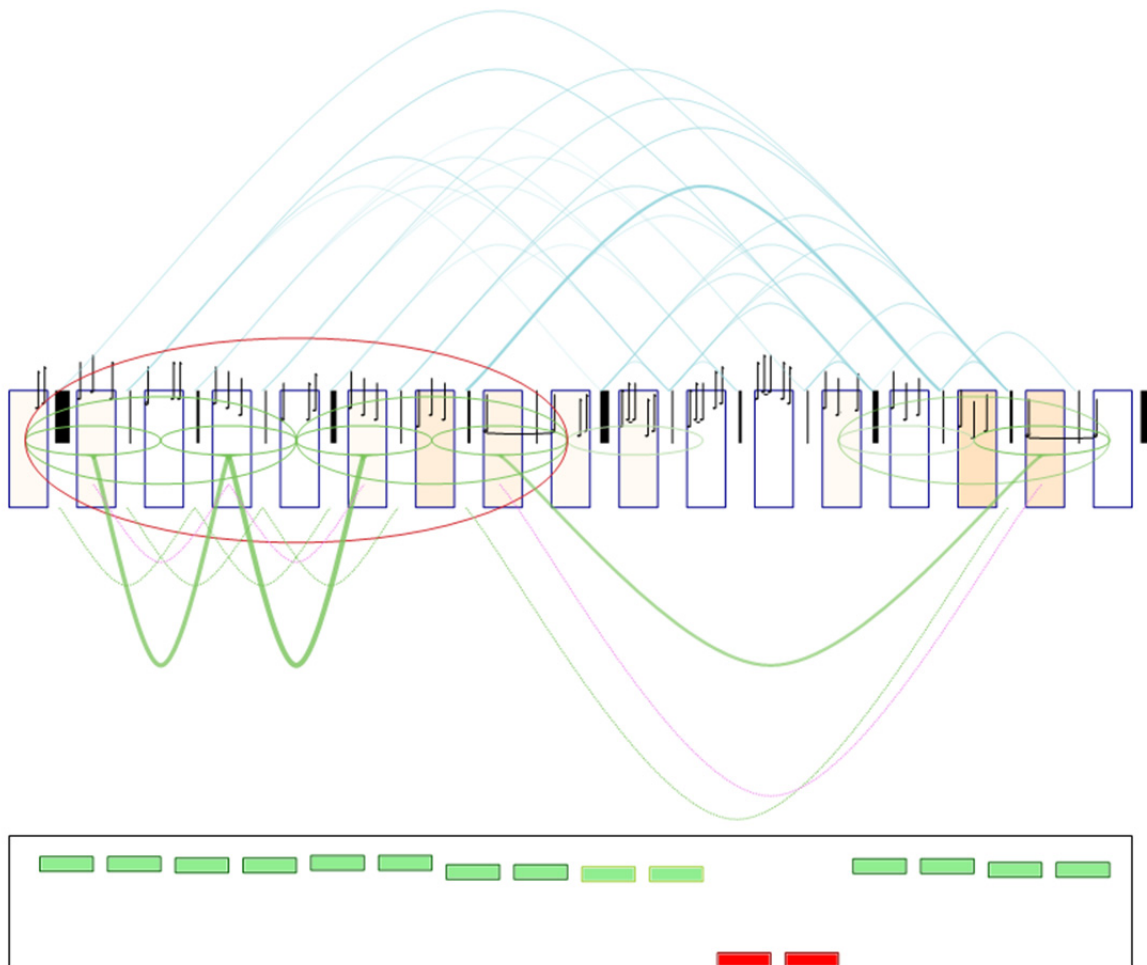


Figure 7.40: Tennessee Waltz, second half (run 2), end of processing.

After a bit more time, the program's listening performance is over. Sadly, the two 4-measure groups in measures 9–16 have disappeared. Musicat is aware (see the red happiness rectangles) that measures 11–12 are missing any group structure. As often happens, it found

a good grouping structure but threw it away in its constant search for something better. This strategy paid off well a few measures earlier in the run, when the strange 7-measure group (7–13) was replaced with the red group (1–8) that persisted all the way to the end of the run, but it did not work so well for the groups (9–12) and (13–16). Several analogies between the first part of the melody and the last part had been formed earlier, but, like these groups, they were also destroyed. One new analogy that was discovered near the end of the run links two phrase endings: (7–8)↔(15–16), just as in the previous run, but there were many more potential relationships that were missing in this run.

What do these runs on “Tennessee Waltz” tell us about the program? In most of the runs (on each of the individual halves of the melody), Musicat eventually settled on an acceptable group structure at the end of the run, but one that was lacking in larger structures. The program did not find enough of the 4- and 8-measure groups that seem quite clear in the structure of this melody. It found several reasonable analogies in each run, but it seems that had it found better grouping structures, it would have found more analogies. When Musicat finds analogies, that fact influences the grouping structure by suggesting possible new groups and by strengthening groups involved in good analogies, but this mutually-reinforcing effect was perhaps not strong enough, in this melody, to help the program quickly settle on a strong perceived grouping structure. If this mechanism worked better and if Musicat more consistently found a very strong set of 4-, 8-, and even 16-measure groups, I suspect it would have a much easier time listening to the entire 32-bar melody and would no longer generate “alien diagrams”. A strong understanding of the first set of 16 measures would make it much easier to understand the second set of 16 measures. Indeed, if the program had settled on its grouping structure for measures 1–16, it could understand measures 17–32 in terms of the first 16 measures, making analogies galore between segments

of the two halves. By the time a human listener has heard measures 1–24 (the **A A' B** parts of the melody), hearing the start of measure 25 makes the final measures, 26–32 almost trivially predictable, because the listener can simply expect to hear another instance of **A** or **A'**.

GOOD PEOPLE ALL (HOFSTADTER)



Figure 7.41: Opening measures of Good People All, from a cantata by Douglas Hofstadter.

This melody is from the piano introduction of the aria Good People All, by Douglas Hofstadter. I have included it in this Complex Melodies chapter because, although it is shorter and perhaps simpler-looking than other melodies in the chapter, it has several features that are especially complex with respect to Musicat’s abilities. In my original work on Musicat I had intended the program to listen to melodies more like this one, whose musical tradition is more of the baroque or classical variety than it is folk or popular. I had also originally planned to use melodies in Steve Larson’s “Seek Well” microdomain, in which all the notes are equal duration. This melody mostly fits in that microdomain: all the notes are eighth notes, except for a few exceptions in the final two measures.

Before looking at Musicat’s listening performances, let’s consider the structure of this melody. I anticipate several significant problems for Musicat that derive from the melody’s relentless stream of eighth notes (a “motoric” rhythm reminiscent of those in Bach’s Inventions.) Musicat’s primary mechanisms for grouping and analogy-making rely heavily on

the music having distinct rhythmic patterns, but nearly every measure here is rhythmically identical. Within this stream of eighth notes, however, there are obvious patterns that evoke more complex rhythms. Most obvious are the descending scale segments in measures 2 and 4. The first note of the start of each descending scale is extremely salient for a human listener, so it sounds like there is an accent on beat 2. In measure 1 (and measure 3), beats 1 and 3 also sound accented because they occur on strong beats and form the start of small groups. Finally, a repeating pattern of notes causes the notes at the end of measure 1 (or measure 3) to sound like they form a group that continues into the next measure, so beat 1 or measure 2 (or measure 4) is not accented. The next figure hopefully makes this implied rhythm (or equivalently, in this case, the implied grouping structure), clear:



Figure 7.42: Good People All (measures 1–4): starting points of groups indicated by upper voice.

In this figure, the upper-voice notes (stems up) indicate the notes I hear as the beginnings of groups. Their durations have been extended to correspond with the lengths of the implied groups. For instance, the high E in measure 2 is a dotted half note, indicating that I hear a group starting on that E that is a dotted half note in length (the 6 eighth notes descending in a scale). To make this even clearer, the following figure shows only the groups' starting notes:



Figure 7.43: Good People All (measures 1–4): groups represented by their starting notes.

The four measures in the figure have a distinctive rhythm when one hears them in this way — they are no longer simply a stream of eighth notes.

Now, remember that the preceding discussion has focused on a human hearing of this melody. Musicat is missing some of the essential abilities that would allow it to hear the melody in this way. Most importantly, recall that Musicat's groups must start on measure boundaries (unless the melody has pickup notes, in which case starting points are systematically shifted). The groups implied by the preceding figure, however, start in a variety of places: beat 1 and beat 3 in the first measure, beat 2 in the second measure, and similarly for measures 3 and 4. Musicat is not able to make these groups; for example, it can make a one-measure group containing measure 2, but not a partial-measure group containing only the 6 eighth notes of the descending scale in measure 2.

Not only does this melody suggest a *small*-scale grouping structure (lengths less than a measure) that Musicat cannot represent, but also the 10-measure length of the melody suggests a *large*-scale group that might pose problems (since 10 measures is longer than the more normal power-of-2 length of 8 measures) might pose problems for the program. (An amusing coincidence involving the number 10 in two different contexts: we might expect a grouping boundary on the bar line after the first 8 eighth-notes, between measures 1 and 2, but a strong group boundary actually occurs 2 notes later, after the first 10 eighth notes of the melody; similarly, we might expect a strong group boundary after measure 8, but then we see that the melody doesn't end until 2 measures later, after measure 10.) To illustrate how the group lengths in this melody — at both small and large scale — differ from the standard group lengths that Musicat expects, I *recomposed* the melody, creating a version with only 8 measures total, and likely group boundaries falling at measure boundaries:



Figure 7.44: Good People All, recomposed.

I claim that much of the interest in the original melody derives from the listener hearing the melody as a surprising variant of a more-expected melody such as my recomposed version. For example, in the recomposed version, the end of measure 1 corresponds to the end of a group, and this ending is expected, because the first two beats have been exactly repeated. In the original version, a group extends from measure 1 through the end of the first beat of measure 2, and this extension is a surprise. Similarly, the leap up to the E on beat 2 in the original melody is more surprising than the leap up to E on the downbeat in the recomposed version. Similar arguments apply at the end of the melody: in the recomposed version, it ends after 8 measures, but in the original version, phrase extension results in the melody continuing all the way to measure 10 before reaching a cadence.

A sophisticated, human-like listening performance for this melody would require the interplay between the expectation for something like my recomposed melody and the actually-heard melody. Tension would be created by groups extending longer than expected or starting on surprising beats. But these are issues for a future version of the program. Let's see what Musicat can "hear" at present in "Good People All".

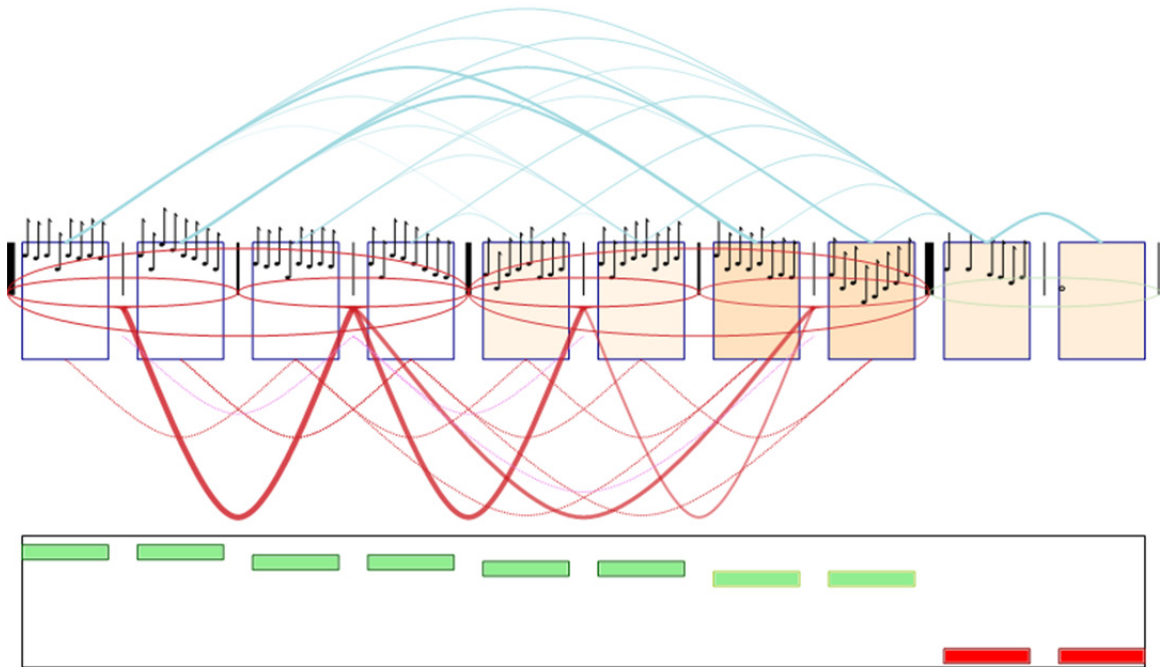


Figure 7.45: Good People All (run 1).

Musicat performed remarkably well, considering its inability to make the smaller groups suggested above, such as the group of 6 eighth notes in the descending-scale in measure 2. Those smaller groups were not (and could not be) formed, but Musicat's smallest group here, (1–2), is a perfectly reasonable-sounding group. The only problem is that it doesn't have any subgroups that would highlight the interesting substructure. I will return to this point in a moment when looking at the analogies, but first, what about the *superstructure*? That is, do the largest groups make sense? The first 4-measure group, (1–4), agrees with my hearing; measure 5 starts a distinct second part of the melody with new melodic material. The second 4-measure group, (5–8), however, doesn't sound like a group to me. Perhaps it would be fine if the group could be extended 1 beat to the right so that it could include the first note in measure 9. However, Musicat's groups can't extend in this way. Additionally, I am unclear about how I hear these measures. I may simply hear a large group that extends all the way to the end of the melody: (5–10). In this run, the final two measures

are separate from the rest of the melody, in a very weak group of their own (the group ellipse is present but difficult to see because the group is so weak). Musicat hears these two measures as a separate group, however, because they have very different rhythms from the other measures in the melody, and also because it (incorrectly) perceives a thick bar line after measure 8. Measure 9 ends with an ascending scale that connects smoothly to measure 9, but the group boundary indicates that Musicat doesn't hear this connection. If the program were able to make neighboring groups that overlap by a single note (group elision), then this listening performance could be made much more plausible by simply extending group (5–8) as mentioned above, and allowing this group to overlap slightly with group (9–10). Regardless, in this run the program unfortunately hasn't seen that there is phrase that extends all the way through measure 10. Even if measures 9 and 10 belong to a separate small group, they should be connected to the larger phrase. All in all, the large-scale grouping structure is not terrible, but it misses the subtleties of the connection between measures 8 and 9 and the extension of the phrase through the end of measure 10.

Returning to the start of the melody and the smaller groups, we see that the strongest analogy created was (1–2)↔(3–4). This analogy is strong because Musicat noticed a contour relationship between the two groups: (1–2) has exactly the same contour as (3–4). The second group is a version of the first that has been transposed down by two steps (with an accidental, G♯, in measure 3 to emphasize the A-minor chord). This contour, importantly, includes the highly-salient descending scale in measures 2 and 4. But Musicat doesn't notice the significance of the scale (in contrast, a previous version of Musicat would almost certainly have made groups containing each of these scales alone — see the next chapter).

Several other analogies were created. The second-strongest was (3–4)↔(5–6), which indicates that Musicat heard (5–6) in terms of (3–4) — another instance, as in some earlier melodies, of the program hearing how initial musical material is developed to yield later

material. In this case, the rhythm of (5–6) was exactly the same as (3–4), but in addition the program heard the contour of these groups as somewhat similar. Two more analogies were created, both involving (7–8), but these were weaker; the program heard these two this group as analogous to earlier groups, but the connection was much weaker than for other groups; I agree that the material in this group seems less related to what came before. And, finally, the last two measures (9–10), are not connected by analogy to any preceding groups, which makes sense considering how different the rhythm is: these are the only measures that are not made of a string of eighth notes. The relative strengths of all these analogies (and the lack of analogies at the end) seem accurate from my own listening perspective.

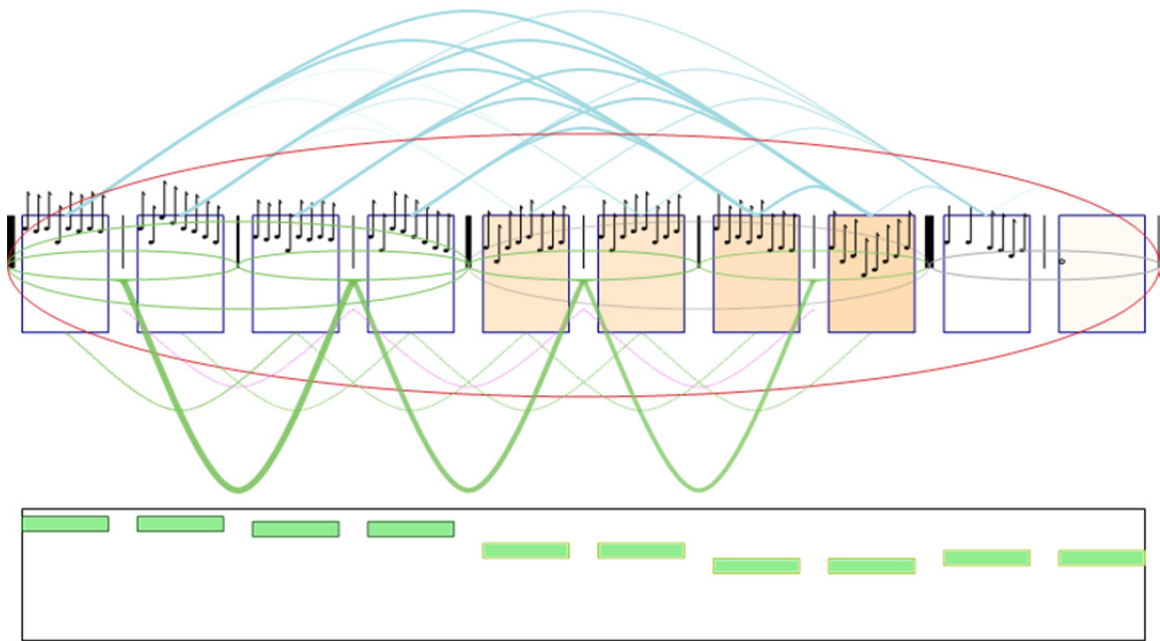


Figure 7.46: Good People All (run 2).

A second run was very similar to the first. One analogy is missing in this picture compared to the previous run, but the most important analogies are present: each 2-measure group is linked to the following 2-measure group, showing that Muscat hears a forward-

development of the melodic material. Again, the analogies are slightly weaker as we proceed from left to right: the exact transposition linking the first two groups is much stronger than the rhythmic repetition and partial contour similarities involved in the later analogies.

This listening performance resulted in all the same groups as the first, with the addition of a 10-measure group enclosing the entire melody. I was happy to see this: it shows that the program has included the final two measures as part of the large structure. I wish that it had heard one additional layer of grouping and added the 6-measure group (5–10), but otherwise the grouping structure is fairly close to how I hear it.

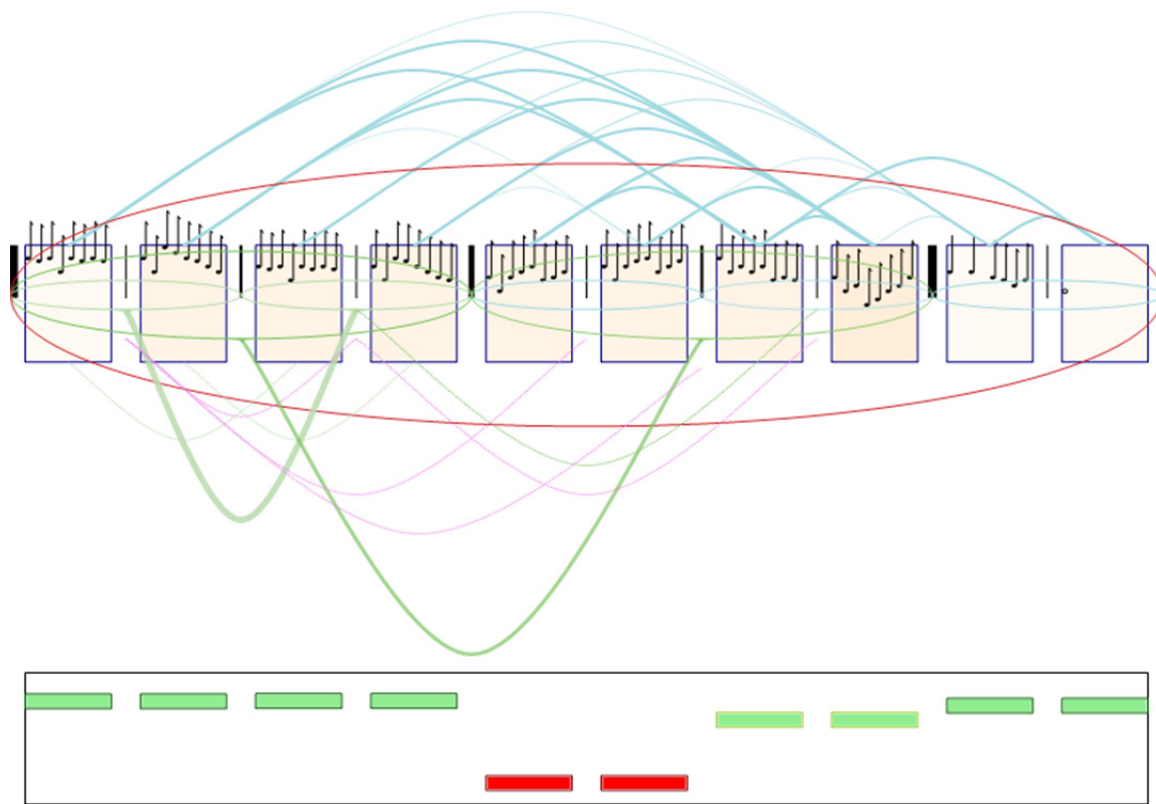


Figure 7.47: Good People All (run 3).

In this final run, the same grouping structure was created as in the previous run. Two of the analogies from the first run are present: the strong analogy (1–2)↔(3–4) as well as the

much weaker and longer-distance analogy $(3-4) \leftrightarrow (7-8)$, which was present in run 1 but missing in run 2. In this run, a larger analogy was formed, linking the two 4-measure groups together: $(1-4) \leftrightarrow (5-8)$. I don't hear this analogy, and it is a little hard to understand. The sub-analogy $(3-4) \leftrightarrow (7-8)$ contributed to this larger analogy, and in addition, Musicat made two rather strange non-rhythmic links (probably contour links, but it is not indicated in the notation): one between the 2-measure group $(1-2)$ and the 4-measure group $(5-8)$, and the other between $(1-2)$ and $(5-6)$. These links involving $(1-2)$ seem strange to me, and helped the program form this larger analogy. All in all, this third run is harder for me to interpret, but it does show quite a bit of consistency with respect the previous two runs.

In summary, these three runs gave remarkably similar results. The grouping structure always had two 4-measure groups followed by a smaller 2-measure group at the very end, and in runs 2 and 3, the entire melody was included in a 10-measure group. The analogies found by the program were similar in the first two runs, whereas in the last run it found a slightly odd larger-scale analogy.

The biggest failings of the program were the ones predicted at the start of this section: the 10-measure length of the melody wasn't handled as well as possible: the 6-measure group $(5-10)$ never formed on any of the runs. This lack of a large 6-measure group structure had the small-scale counterpart of missing 6-eighth note groups in measure 2 and 4 — no surprise, since Musicat's architecture doesn't allow this sort of group! But the lack of these small-scale groups was somewhat covered up by the strong 2-measure grouping structures that enclosed each of the areas where we would like to see sub-groups. Even though the smallest groups could not be formed, analogies involving the 2-measure groups were formed, and these analogies matched my own hearing of the melody quite well. Overall, the program

did better than I expected, given its inability to hear the smallest groups and the homogenous nature of the eighth-note rhythm.

SUN AND MOON (BOUBLIL AND SCHÖNBERG)

Kim:

You are_ sun - light and I moon, joined by_ the gods of for - tune, mid - night and

7 high noon shar - ing_ the sky. We have been blessed, you_ and I.

13 Chris:

19 You are_ here like_ a mys - t'ry._ I'm from a world that's so dif - f'rent from

all that_ you are. How in_ the light of_ one night did_ we come so far?

Figure 7.48: Sun and Moon (from the musical *Miss Saigon*).

The song “Sun and Moon” is the final melody in this chapter, and also was the last melody I decided to add; I didn’t even think of including it until long after I had stopped modifying the program code. Along with the Bad Melodies (thanks to their random nature), and a late addition to the Simple Melodies section, “Frère Jacques”, it is therefore one of the most “pure” tests of Musicat’s listening abilities; nothing in the program’s design was influenced by the knowledge that it would be listening to this melody. This will likely be obvious in its listening performances: I know *a priori* that there are many things about this melody that Musicat will be “deaf” to, and thus this melody should illustrate directions for future improvement.

“Sun and Moon” is one of my favorite songs from my favorite piece of musical theatre, *Miss Saigon*, composed by Claude-Michel Schönberg (a distant relative of Arnold

Schönberg), with lyrics by Alain Boublil. Since I have heard this piece — and the entire musical it is a part of — countless times, my own experience of listening to the song is deeply influenced by my memory. As I mentioned when describing Bad Melody 5, Musicat has no long-term memory of this sort, so it misses this part of the listening experience. For me, the most salient feature of the melody is the rhythmic figure in the first two measures. Measure 2 is a development of the rhythm in measure 1 — it simply is a version of measure 1 where the second note has been shortened and an additional quarter note has been added at the end of the measure (this same type of thing happened at the start of “Row, Row, Row Your Boat”). But more importantly, measure 2 is a restatement of the most important rhythmic figure in the entire two-hour score of *Miss Saigon*. This rhythm appears directly in the melody of “Now that I’ve Seen Her”, and in the orchestral accompaniment to the songs “Last Night of the World”, “Telephone Sequence”, “I Still Believe”, “The Fall of Saigon”, “Ellen and Chris”, and the song “Too Much for One Heart” (which was cut from the production). It also appears prominently in diminution (*i.e.*, the notes’ durations have all been cut in half) in the very first notes of the whole score (“Opening Act One”):



Figure 7.49: *Miss Saigon*, opening notes.

The rhythm in question is evident when we listen to the accented notes. If we extract the accented notes and rewrite them as dotted quarters and quarters with a doubled tempo, the relationship to measure 2 in “Sun and Moon” is obvious:



Figure 7.50: *Miss Saigon*, opening accented notes, rewritten at double tempo.

Notice that the accent marks above were not strictly necessary — each of these notes occurs at a local high point in the melody, and it would thus sound somewhat accented without any deliberate articulation by a musician performing the piece. But in any case, Musicat currently notices only rhythms that *appear* in a score as notes with various durations, not those *implied* by accented notes.

The context provided by all these other examples of this rhythmic motif is, to me, *essential* in how I hear “Sun and Moon”. Musicat will likely hear measure 2 as a rhythmic variant of measure 1 (a truly A. Schönbergian way of hearing this C.M. Schönberg melody!), and also will notice instances of these rhythms that are later repeated, but unfortunately cannot understand the significance and omnipresence of the measure 2 rhythm. By contrast, a human being listening to a production of *Miss Saigon* will practically be beaten over the head with this “dotted-quarter, dotted-quarter, quarter” figure for two hours, and would certainly notice its significance and the coherence it provides to the work.

Another aspect to my personal listening performance of this song is my understanding of the lyrics. These are completely unavailable to Musicat, by design, but it is helpful to remember that Musicat is missing out, through no fault of its own! Lyrics have the potential to influence our hearing of the grouping structure of a piece. If we consider the lyrics alone (paying attention to punctuation and rhyme), we may infer the following grouping structure (groups are indicated with phrase markings above the staff):

Kim:

You are_ sun - light and I moon, joined by_ the gods of for - tune, mid - night and

7 high noon shar - ing_ the sky. We have been blessed, you_ and I.

13 Chris:

You are_ here like_ a mys - t'ry._ I'm from a world that's so dif - f'rent from

19 all that_ you are. How in_ the light of_ one night did_ we come so far?

Figure 7.51: Sun and Moon, with phrasing implied by the text.

Another possible way of hearing the grouping structure, however, is suggested by considering the harmonic structure of the piece. While the chords are not shown in the following figure, I claim that this is a reasonable grouping structure:

Kim:

You are_ sun - light and I moon, joined by_ the gods of for - tune, mid - night and

7 high noon shar - ing_ the sky. We have been blessed, you_ and I.

13 Chris:

You are_ here like_ a mys - t'ry._ I'm from a world that's so dif - f'rent from

19 all that_ you are. How in_ the light of_ one night did_ we come so far?

Figure 7.52: Sun and Moon, with an alternate possible grouping suggested by the chords.

I will not argue for the “correctness” of either of these possible ways of hearing the grouping in this melody — both have some merit and might be reasonable listening

performances (likely influenced by the particular stage performance heard by the listener). Regardless of the grouping structure that is heard, there are some phrases consisting of an odd number of measures. In the first half of the melody (measures 1–12), both the lyric-based and chord-based grouping structures suggest the 3-measure group (10–12). In the lyric-based analysis we also have the group (1–3), while in the chord-based analysis we have the group (7–9). And at the very end of the melody, measures 21–25 form a 5-measure group. For this melody, then, finding one or more groups having an odd number of measures seems crucial for a cogent listening performance. We already saw in Bad Melody 4 that Musicat has trouble finding odd-length groups, so how will it hear this melody? (Keep in mind that the program has no knowledge of the lyrics or the chords.) In the runs that follow, I restricted the input melody to half the length of the melody at a time; first we consider measures 1–12.

Sun and Moon, First Half

Kim:

You are_ sun - light and I moon, joined by_ the gods of for - tune, mid - night and

7

high noon shar - ing_ the sky. We have been blessed, you_ and I.

Figure 7.53: Sun and Moon, first half.

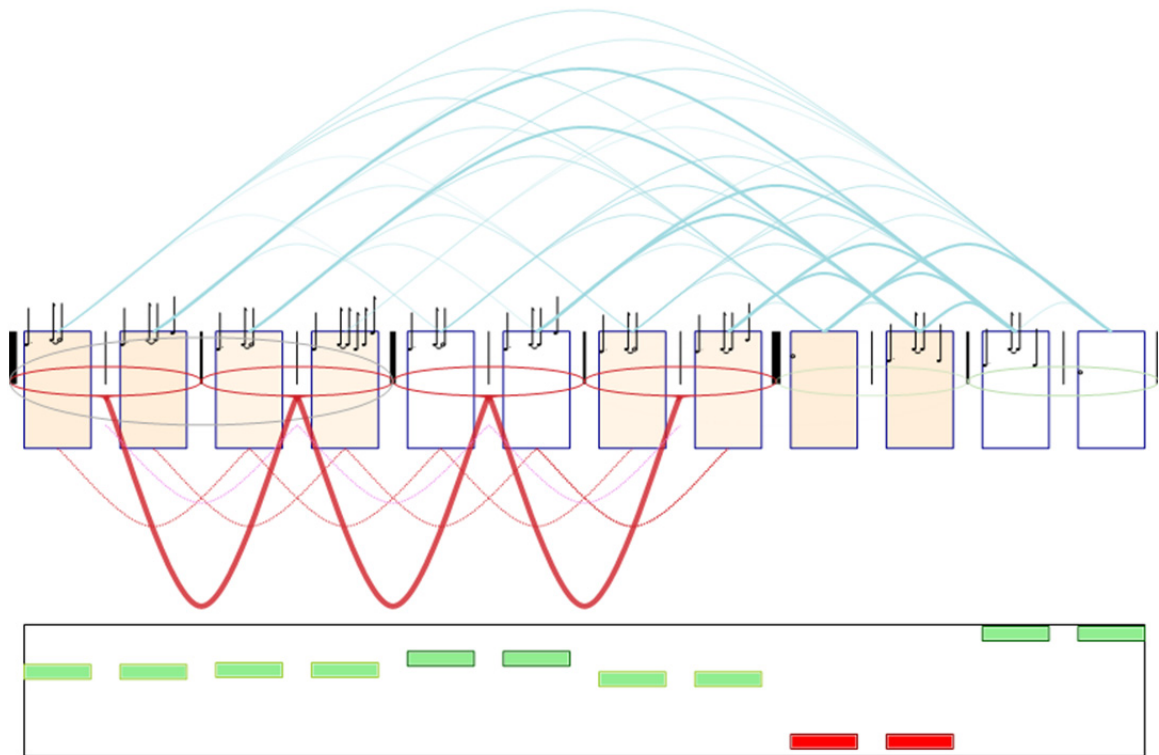


Figure 7.54: Sun and Moon, first half (run 1).

This run seems to have started well: the first six measures are grouped into three red groups of two measures each. This grouping into pairs is consistent with the chord-based grouping suggested above, and also consistent with the observation that measure 2 sounds like a slight modification of measure 1. Musicat has also made groups of the measure pairs 3–4, 5–6, and 7–8. This grouping makes sense because the second measure of each pair can be heard as a slight variant of the pair’s first measure, and Musicat tends to group together adjacent structures when they are similar to each other (grouping and analogy-making¹⁰ are two very different activities, but when analogous structures are adjacent, they are also good candidates for larger-scale grouping). This observation about measure-pairs applies not only

¹⁰ By “analogy-making” I intend to include Musicat’s low-level ability to notice measure similarity. Even though measure-to-measure connections based on rhythmic or pitch similarity are not official “analogy” structures for Musicat, I include them in the grouping heuristic discussed here.

the rhythms but also to the pitches of these measures: the first measure of each group has exactly the same first two notes as the measure that follows. The red analogies between successive groups also make intuitive sense: each 2-measure group consists of a first measure that has only two notes, while the second measure of each group is an elaboration with additional notes at the end; Musicat notices the similar structure of these groups.

The 4-measure group (1–4) is less understandable to me: the end of measure 4 features a fast rhythm that moves the melody forward, connecting it with measure 5 and making it hard for me to hear a higher-level grouping boundary between these measures. We can see by the thin ellipse in the previous figure that the group is, fortunately, quite weak. More disappointing is the presence of group (7–8), where I had heard the longer group (7–9) instead. During the run, the presence of the short group (7–8) would be expected and even encouraged at first; it would simply continue the pattern established by the previous 6 measures. Indeed, after measure 6, the (7–8) group was present as a purple expectation:

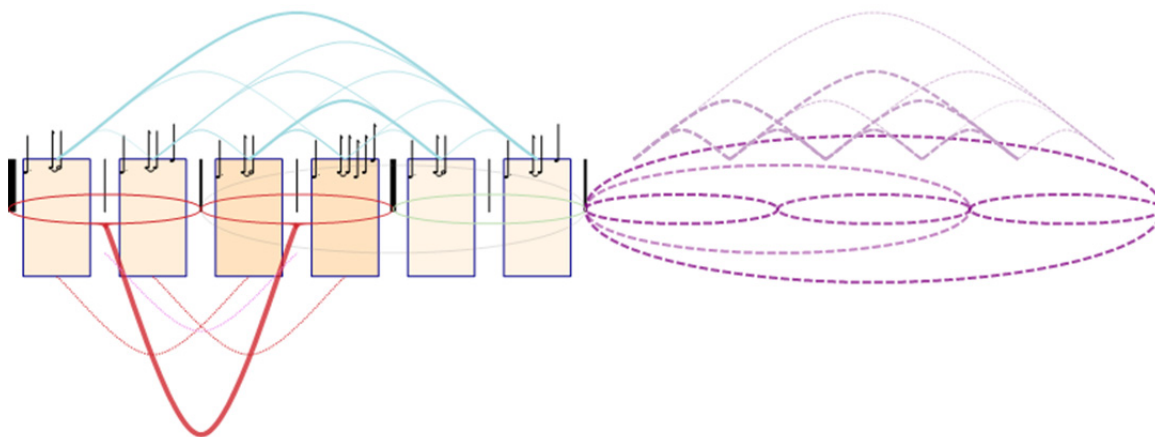


Figure 7.55: Sun and Moon, first half (run 1), after measure 6.

But after measure 8 arrived and the expected group (7–8) was formed, a new expectation for group (9–10) formed. Skipping forward several measures until after measures

9 and 10 arrived, we see that this expectation was still present, although 9–10 remained ungrouped:

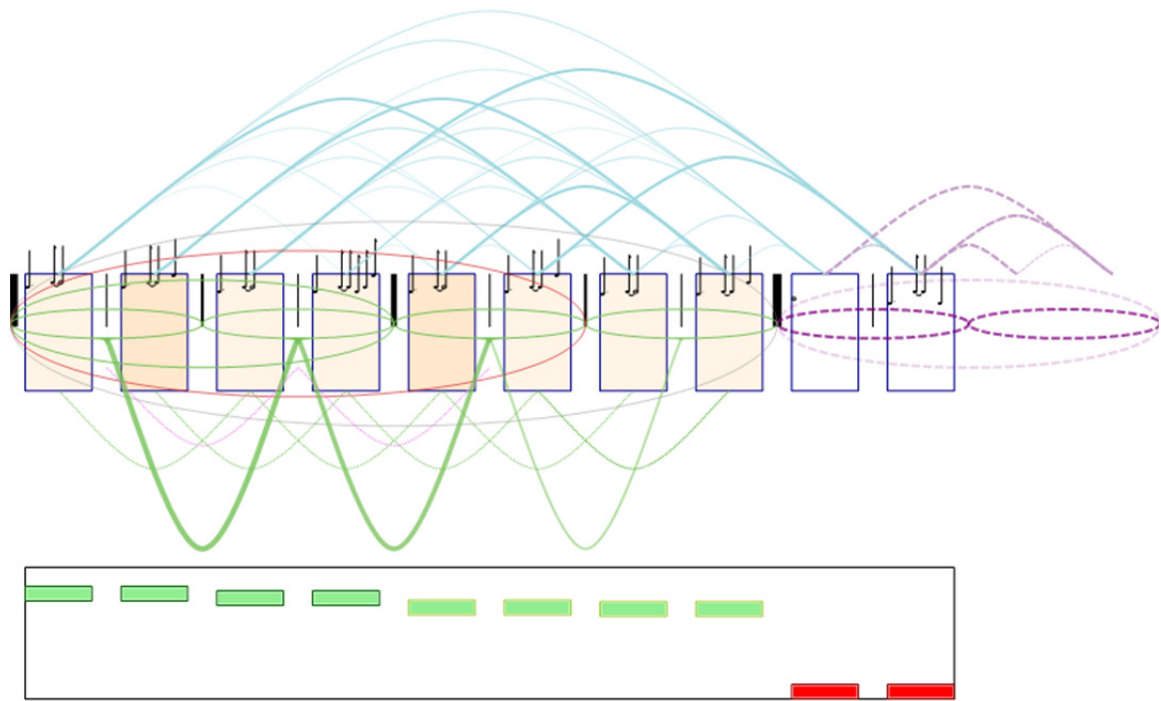


Figure 7.56: *Sun and Moon*, first half (run 1), after measure 10.

Here we can see that Musicat is unhappy with the situation in the final two measures. We can see three developments that seem to be preventing the formation of the desired group (7–9), however. First, group (7–8) is doing fine on its own, as far as the program is concerned. Musicat has found a slightly weak analogy, but an analogy nonetheless, between it and the previous 2-measure group. Second, (7–8) is also contained within a much larger group, (1–8), which is weak, but is nonetheless serving to insulate the group (7–8) somewhat from the following notes (in order to extend group (7–8) to the right, Musicat would first have to break group (1–8), to “free up” its child groups.) Third, there is a very thick bar line standing between measures 8 and 9, serving as a roadblock, or worse, a chasm

discouraging the making of groups that straddle the gap. Of course it is possible for groups to straddle such boundaries (see the red group (1–6) for an example), but boundaries make this much more difficult. Finally, Musicat doesn't see any relation between measures 7–8 and measure 9; no reason has emerged to join these into a 3-measure group. Indeed, it's a good question: why *should* they be grouped together?

Two good reasons jump to mind. The first is one that Musicat is aware of: the whole note in measure 9 is practically screaming out to be heard as the end of a group, simply because of its long duration. The importance of long time intervals between successive note onsets in establishing group structure has been well established in other models of music cognition, and plays a role in Musicat as well. Unfortunately, this pressure for the whole note to be the *end* of a group was not strong enough to overcome the forces exerted by the other pressures mentioned above that serve to maintain the status quo (*i.e.*, the happy status of the shorter group (7–8)) and the thick bar line that indicates that measure 9 should be the *start* of a group.

The second good reason for hearing a group that extends through measure 9 is one that Musicat is not aware of in its current incarnation: the note A in measure 9 is a continuation of a pattern of downward melodic motion present at a higher level of analysis (in a Schenkerian sense). The melody starts on D in measure 1 and continues down all the way to the D an octave lower in measure 12:



Figure 7.57: Sun and Moon, measures 1-12. Descending scale shown with large noteheads.

This pattern looks much like a Schenkerian *Urlinie*, although it doesn't qualify as such according to Schenker's strict definition requiring motion starting on scale degree 3, 5,

or 8, since this passage starts and ends on scale degree 2 and skips the note E entirely. In any case, the downward motion is very salient, and, importantly, measures 7 and 8 both start with the leading tone, B, which sounds unstable. The A in measure 9 provides a much more stable point of repose. (The program does not have access to the chord structure in the musical score, but that, too, would corroborate this analysis: the first beat in each of measures 7 and 8 is harmonized with the minor iii chord, E minor, while measure 9 has a more stable-sounding IV chord, F major.) Because of the long duration of the note A, measure 9 sounds very obviously (to human ears) like the end of a group. Musicat is aware of note stability, but in this example the important issue, with respect to grouping, is the stability of only those notes that are part of the high-level linear-descent structure. We need to focus on the notes B and A in these measures, temporarily ignoring the other notes, in order to hear how the linear structure and note stability strengthen this grouping choice. (Larson's multi-level Seek Well model, discussed in Chapter 2, deals with these sorts of issues, whereas Musicat, in its present state, does not.)

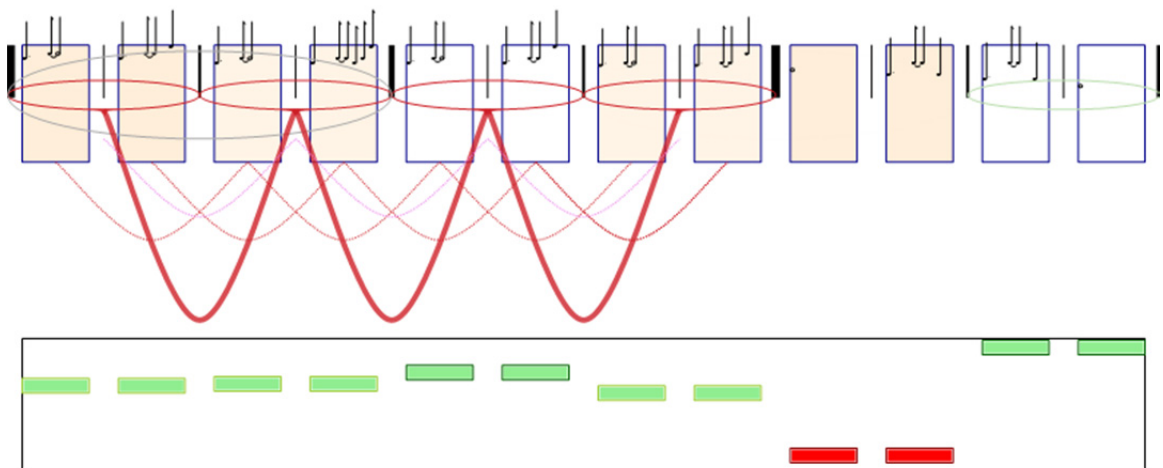


Figure 7.58: *Sun and Moon*, first half (run 1), medium detail.

At the end of the run, the program remains unhappy with measures 9–10, but unfortunately it never resolved the problem; the program seems confused by the final 4 measures. Although it seemed unlikely that things would get better, I tried several more runs just in case.

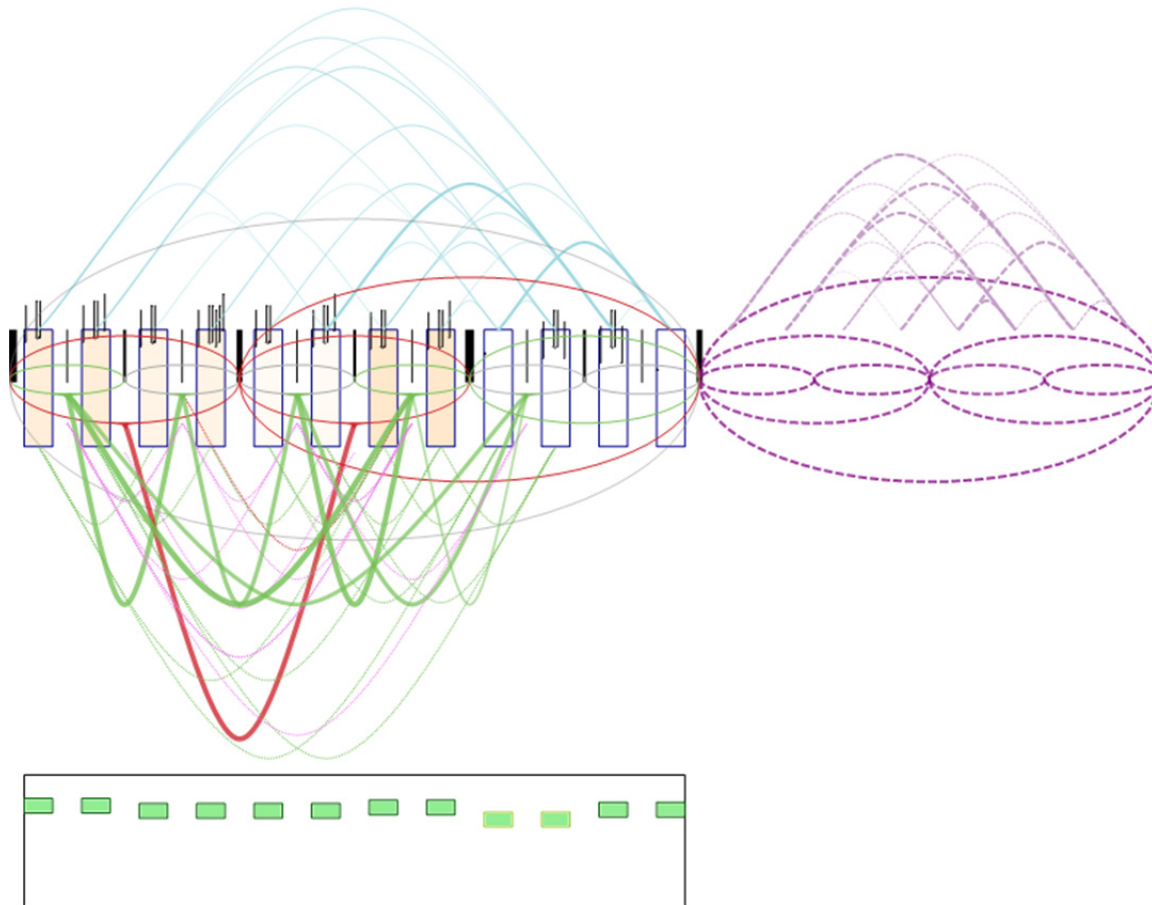


Figure 7.59: Sun and Moon, first half (run 2) (final processing).

In a second run, the grouping structure was still a problem. Larger meta-groups were formed, combining the shorter 2-measure and even 4-measure groups in not-too-unreasonable ways, although these larger groups are hard to make sense of since the lower-level groups inside them seem wrong. In this run, many more analogies were discovered. The

four successive green analogies from run 1 are present, and in this run many more green analogies were also formed, linking distant groups instead of just neighbor groups. A large red analogy was also formed during the final listening stage, although it disappeared at the very end of the run. The next figure shows the final state:

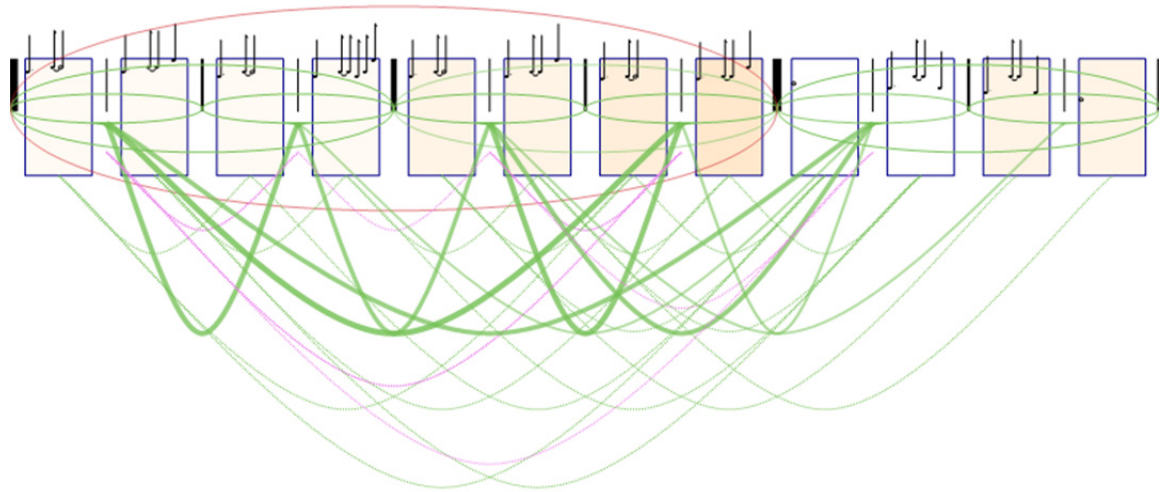


Figure 7.60: Sun and Moon, first half (run 2), medium detail.

There are many green analogies, and this is unsurprising because each of the two measure groups in measures 1–8 is so similar in terms of rhythm and melodic contour. The final 4 measures have a few analogies to earlier structures, but overall they seem not as strong; the presence of the whole notes in those final measures seem to confuse the program. (If the whole notes were heard as the final notes of 3-measure groups, the analogies involved would be much clearer, I believe.)

A third run turned out similar to the previous ones, so it is not shown here. The 3-measure groups never formed. Now let's turn to the second half of this melody.

Sun and Moon, Second Half

Chris:

You are_ here like_ a mys - t'ry._ I'm from a world that's so dif - frent from

all that_ you are. How in__ the light of__ one night did__ we come so far?

Figure 7.61: Sun and Moon, second half (measures 13–25, renumbered as 1–13 for simplicity).

Notice that I will refer to the measures in this second half as measures 1–13, for ease in counting. This half of the melody is best grouped, in my own listening, using a perfectly regular binary grouping structure for the first 8 measures, followed by the 5-measure group (9–13), or else a 3-measure group, (9–11), followed by a 2-measure group, (12–13), and then a meta-group, (9–13), containing both of these groups. Since Musicat didn't recognize any 3-measure groups in the first half of the melody, I don't expect odd-length groups in this half either, but I wanted to give it a chance.

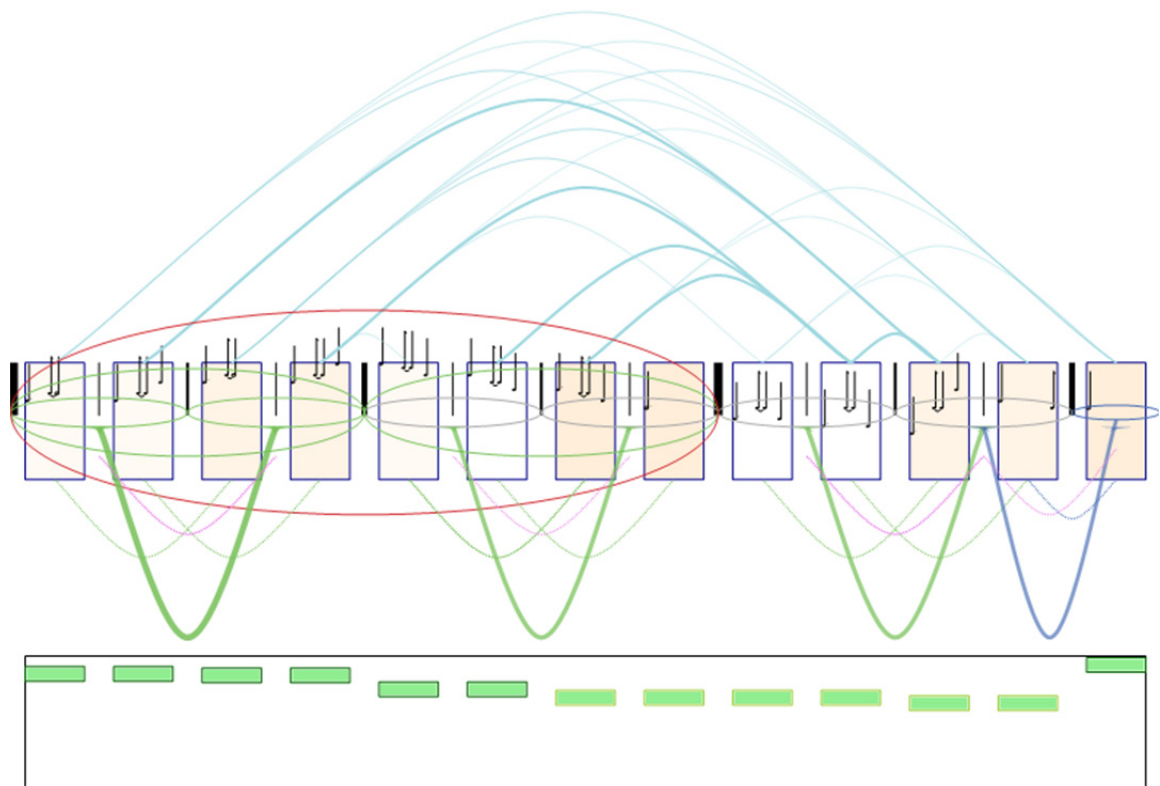


Figure 7.62: Sun and Moon, second half (run 1).

As I had suspected, no 3- or 5-measure groups formed here, but there was a slight surprise: one odd-length group did form — a 1-measure group, (13)! Why did this happen? It turns out that before the last measure was heard, Musicat had generated an expectation for another 2-measure group, (13–14). When measure 13 arrived, it became part of the expected group, but the rest of the expected group never came. However, this nascent 1-measure group ended up getting linked to group (11–12) by an analogy, $(11-12) \leftrightarrow (13)$, and the strength of this analogy helped support the single-measure group (otherwise, it would likely have been destroyed). The analogy was strong-enough to form because Musicat saw measures 12 and 13 as rhythmically similar. Additionally, measure 13 consists of the stable tonic note C, and Musicat gives a strength bonus to analogies in which the right-hand side has a more stable

ending than the left, because this is sometimes a clue to an antecedent→consequent relationship between the two sides.

While I was glad that the final measure participated in some sort of group, the grouping structure of the final 5 measures was still disappointing. Measures 9–11, in particular, form something like a sequence, although it is not a strict sequence of the type Musicat can detect. If we consider only the *first* note in each of these measures, we see a descending sequence (A–G–F). The rest of the notes in each measure, however, behave differently, and do not move down by transposition along with the first notes of the sequence. This passage, then, is again one that suggests the utility of a Schenker-type extraction of a higher-level structure, just as I referred to in discussing the first half of the melody.

On a positive note, however, Musicat did form the expected grouping structure for the first 8 measures of this melody. There are small green groups of 2 and 4 measures, all enclosed in a red 8-measure meta-group. Two green analogies have formed that seem reasonable, although it would have been nice to see a larger-scale analogy such as (1–4)↔(5–8); after all, the initial melodic ascent for 4 measures, followed by a 4-measure descent, is reminiscent of the ascent–descent pair of groups at the start of “On the Street Where You Live”, so I expected this analogy to form. And in fact, it *did* form, momentarily, during the run...

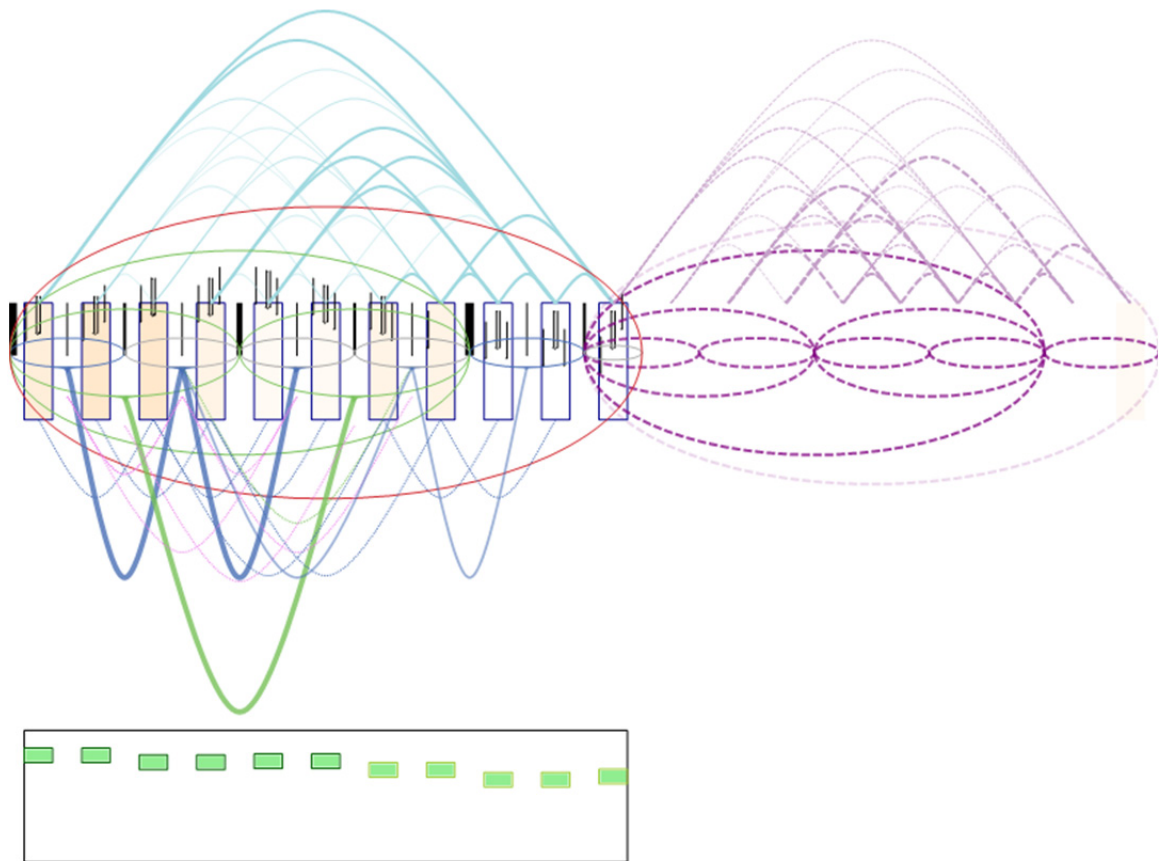


Figure 7.63: Sun and Moon, second half (run 1), after measure 11.

... but unfortunately, it disappeared shortly after it was created. I tried several more runs, hoping for a possible 3- or 5-measure group to be found at the end of the melody or for the larger analogy (the green one in the figure above) to persist to the end of the run.

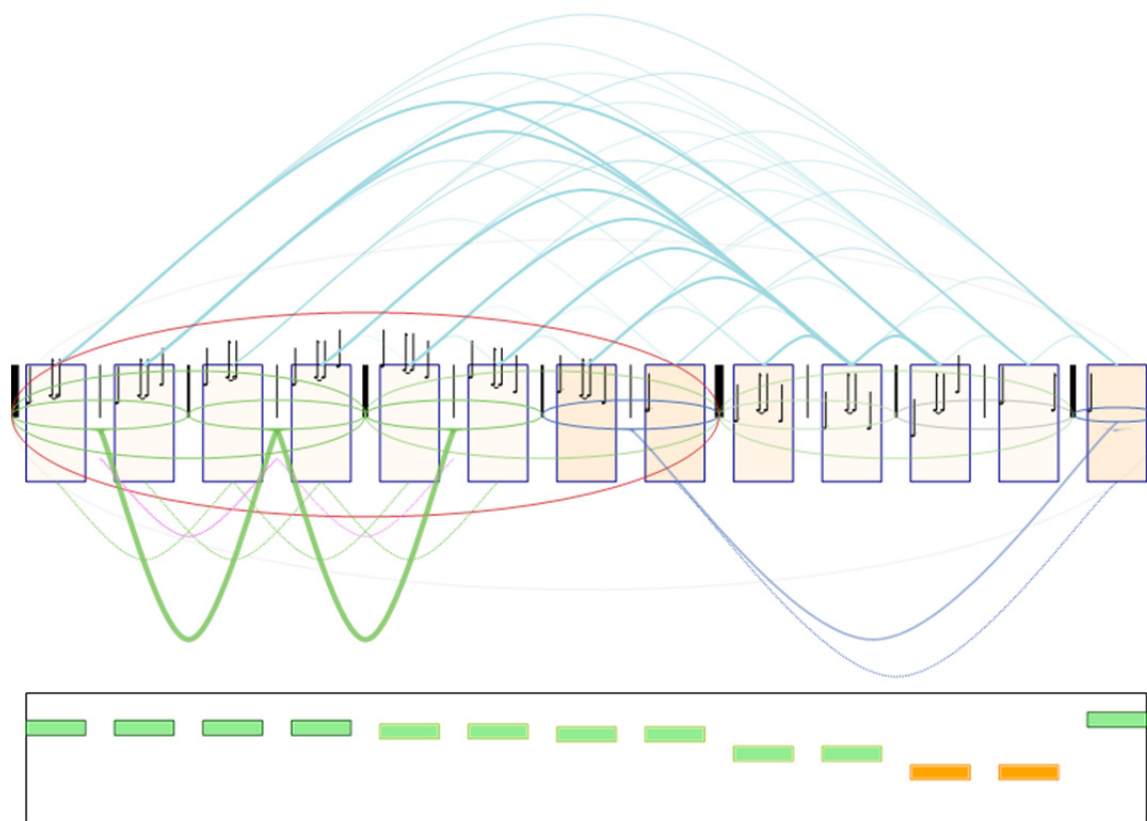


Figure 7.64: Sun and Moon, second half (run 2).

On a second run, the grouping structure of the first 8 measures was the same, but an analogy linking the *second* ascending group to the *first* descending group appeared this time: $(3-4) \leftrightarrow (5-6)$. Unfortunately, the large analogy did not form at all in this run.

In the final measures, however, we see a different picture than last time. The meta-group $(9-12)$ has formed, which is a reasonable group (although I wish Musicat were able to hear it as something like a sequence, even though I know it's not possible with the current code). The final measure 13 has again been perceived as a 1-measure group, but this time it is part of the analogy $(7-8) \leftrightarrow (13)$, which is more interesting than the analogy $(11-12) \leftrightarrow (13)$ in the previous run. Measures 7-8 sound like the end of a phrase, and measure 13 does as well. Of course, if measure 13 were part of the hypothetical group $(12-13)$, this would be an

even stronger connection. Still, it's encouraging that the program heard an analogy between these two groups, despite their different lengths. Ideally, this analogy would have been a strong enough force to make Musicat shift the position of the thick bar line to fall *after* measure 13, instead of before it. This, in turn, might have helped the program to make sense of the grouping structure of those final measures.

One final positive feature of this run is the structure of the blue measure links. Recall that these links fade over time as the program focuses more on current and very recent measures. Thus, in a picture of the program's final state, links between two early measures usually will have faded out, although links involving at least one recent measure can remain strong. In this run, this has occurred as always, but we see a plethora of links from the recent measures extending back in time to previous measures, forming a distinctive shape when we look at links emanating from measure 10 and linking back to earlier measures. What does this shape indicate? It shows simply that Musicat has heard measure 10 as rhythmically-similar to almost every preceding measure! (Why measure 11 doesn't have just as many links is simply a feature of the stochastic nature of the program; for some reason, Musicat paid more attention to measure 10 during this run.) This large collection of links is gratifying because, as I mentioned earlier, the rhythm here (the one in measure 10, and also in measure 9 and 11 and many others) is one of the characteristic rhythms in the whole score to *Miss Saigon*; it's encouraging that the program has noticed all these similarities in this single 13-measure segment.

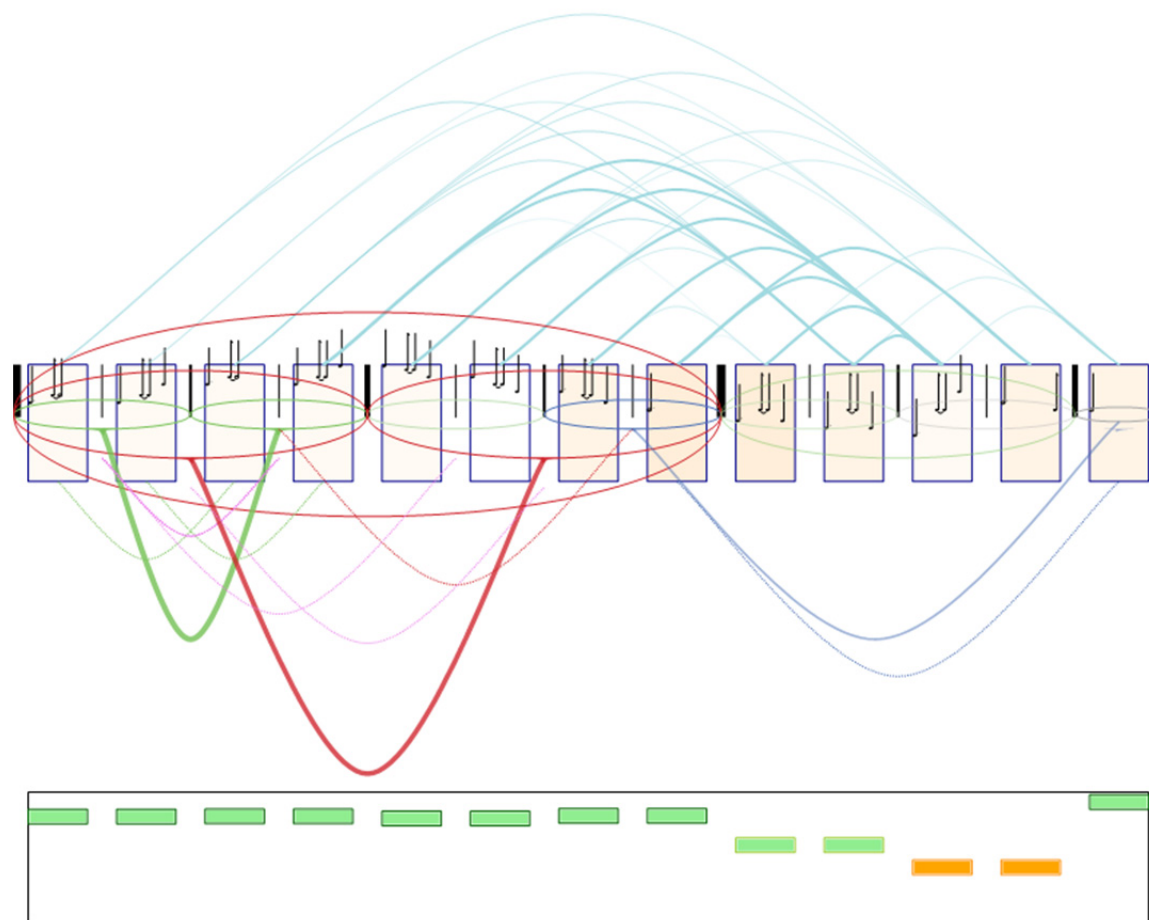


Figure 7.65: Sun and Moon, second half (run 3).

This is a final run on the same melody. Although the program never found the 3- or 5-measure groups I hoped for, this single run contains many of the best features from the previous two runs, and it finally did create and stick with the large red analogy (1–4)↔(5–8). The 1-measure group formed again, along with the link to the other phrase ending, (7–8). Finally, in this run we see even more of the blue measure links connecting early measures to the measures in the range 9–11, all of which exhibit the characteristic rhythm discussed earlier. In short, Musicat has heard many of the things I hear in this melody, but it has totally missed the higher-level linear pattern that makes measures 9–11 sound something like a sequence, and it has not heard the final 5 measures as part of a single group (or meta-group).

CHAPTER EIGHT

The Architecture of Musicat

Previous chapters have described, at a high level, some of the particular aspects of musical listening that Musicat is intended to model. This chapter presents the architecture of the system and describes its inner workings in some detail. The ultimate reference for how Musicat works is the even lower level of the C# source code itself¹¹, but as a cognitive scientist, I generally find the ideas behind Musicat to be more interesting than the technical details of its implementation. This chapter aims for a level of detail somewhere above the level of source code while still providing enough information for someone to try implementing a similar system based on these ideas.

Optimistically, I think that some of the lowest-level details do not matter — I had to make many rather arbitrary decisions along the way when implementing my ideas, and I believe that many of them are not critical, in comparison with overall organization of the program. For instance, I chose a default “urgency” value for many codelets to be 20 (out of 100). But why not 30, or 10, or even 25, 21, or 20.17? Musicat has a large number of such parameters, not to mention other small decisions about how each individual codelet works.

¹¹ Musicat is written in the programming language C#, pronounced “C sharp”. This language does not inherently have anything to do with music — the fact that I used a programming language named after a musical note to write a program that models music listening is simply an amusing coincidence.

Sometimes it bothers me that I have had to make so many decisions during implementation, but I have taken comfort in the fact that other FARG programs such as Copycat have worked fine, with their similarly large numbers of parameters. Another surprising source of optimism was a brief discussion I had with Doug Lenat, when I asked him about choosing parameters in his Automated Mathematician (AM) program. He told me that he simply chose reasonable-sounding values for various weights and other parameters, and AM essentially functioned in the same manner across a range of parameter values (Lenat 2009). I suspect that this is also the case for Musicat, especially considering its stochastic architecture. Its overall plan of attack for making sense of music is critical, but hopefully the way it “listens” is stable with respect to minor differences in implementation.

Overview: Mental Processes to Model

Before describing the architecture, I first review some of the key mental processes Musicat is intended to model: internal representation of music, the flow of time, importance of rhythmic patterns, grouping structure, tension and resolution, tonal pitch structure, expectation, and last but of course not least, musical analogy-making.

- **Internal representation:** music is represented in human minds in a high-level way that throws out the raw perceptual details of incoming sound waves but maintains higher-level features in memory. Musicat is given music at the note level (which is already quite high-level compared with sound waves) and it generates structured representations of what it “hears”. These representations, and the way they evolve in real-time, constitute the most significant “output” of Musicat (as seen in Chapter 5). Even though they

represent *internal* cognition, the groups and analogies formed by Musicat are the objects of interest Musicat makes available *externally* for us to observe.

- **Flow of time:** music, in contrast to some other arts, is critically dependent on the passing of time. The human short-term memory system, in particular, places constraints on how we can hear and understand music. Musicat is provided notes one at a time in simulated real-time, and is limited in its ability to reinterpret notes and phrases that have faded into the past, outside the focus of short-term memory.
- **Rhythmic patterns:** rhythm is a particularly primal aspect of music, and likely contributes more to recognition of musical patterns than pitch (melodies are often recognizable by their rhythmic pattern alone, whereas people have more difficulty recognizing a pitch contour devoid of rhythm).
- **Grouping structure:** Gestalt grouping principles apply to music listening, causing notes to be perceptually grouped in much the same way as objects are grouped together in visual perception. Grouping in music is hierarchical and helps people understand chunked larger-scale structures that would otherwise exceed working-memory capacity.
- **Tension and resolution:** one of the strong clues to grouping structure in music, as well as an important way in which music evokes emotions, is the frequent increase and subsequent relaxation of musical tension. For example, cadences typically involve harmonic tension followed by relaxation to a stable chord, and entire phrases often take the shape of an arch: pitches rise, increasing tension, and then relax via downward motion at the end of a phrase. Musicat uses tension and resolution of various parameters as clues to grouping structure.

- **Tonal pitch structure:** pitch contour is important to melody understanding, and in Western tonal music in particular, the qualities of individual pitches in a harmonic context are quite salient. In Musicat, tonal structure is modeled through a combination of pitch contour and analysis of the stability or the tension inherent in pitches in tonal contexts.
- **Expectation:** grouping, and in general the understanding of rhythmic structure, is aided by expectation. For example, if an eight-measure phrase is established, listeners generally expect the next phrase also to span eight measures. These expectations help guide Musicat's generation of groups.
- **Analogy-making:** as discussed in previous chapters, musical perception (like the rest of perception) is critically dependent on analogies at many different levels. Analogies drive a great deal of Musicat's top-down processing.

Major Components

USER INTERFACE

Because the main output of Musicat is the program's internal representation of what it has "heard", the user interface is a significant component of the program. Two versions of the user interface were developed: a stand-alone program (Figure 8.1) and an "add-in" to the Visual Studio 2010 development environment for rapid feedback during development (Figure 8.2).

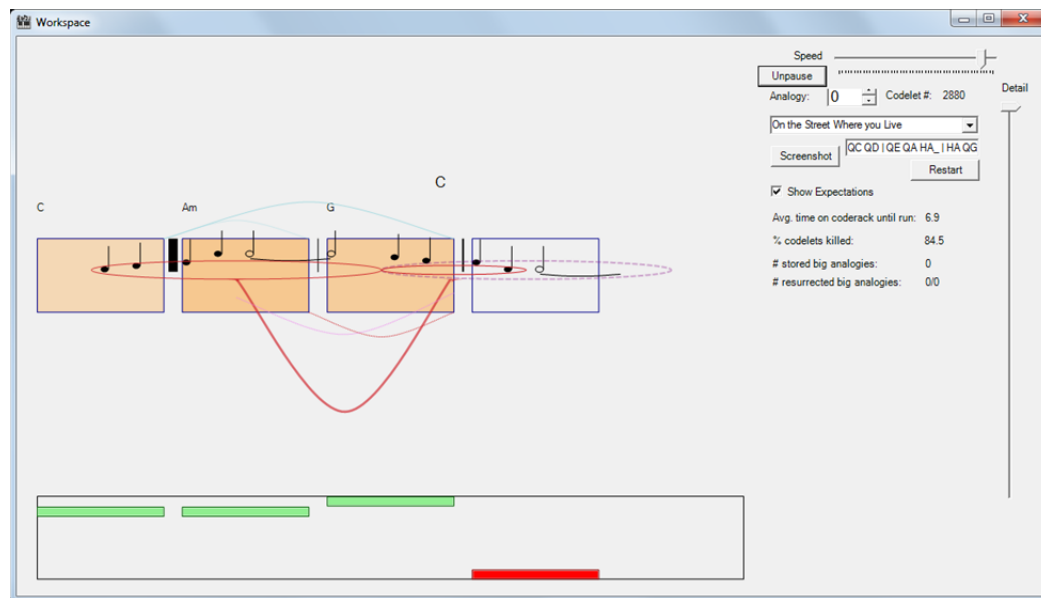


Figure 8.1: Standard user interface for Musicat.

Standard Interface for Musicat

Earlier chapters explained how to interpret Musicat's output. In this section I discuss the input to the program. The primary user-provided input is the melody, represented using a simple text format. A set of characters such as QG#5 represents a single note (in this case, a quarter note on the pitch G#, in octave 5). Notes are separated by spaces, and measures are separated by a vertical bar. Each note is composed in the following manner (optional elements shown in square brackets):

Duration	Pitch	[Octave]	[Tie?]
(S/E/Q/H/W) [.]	(A-G) [b/#]	[0-9]	[_]

The duration symbols represent sixteenth, eighth, quarter, half, and whole notes, respectively. The optional dot after the duration is used to represent a dotted eighth, quarter, or half note. Pitch characters may be followed by a 'b' or '#' to indicate flat or sharp notes. Note that when a note is initially provided to the program, its pitch is represented *internally* simply as

an integer from 1–127, where each number corresponds to a different piano key¹², just as in the Musical Instrument Digital Interface (MIDI) standard. Importantly, the accidentals included in the input are used solely to specify the MIDI number, which means that “C#” and “Db” are equivalent inputs to the program, as they both are indicated by MIDI number 61 (in the middle octave of the piano). The octave number above specifies an octave in the same manner as in MIDI, so that C4 is Middle C (MIDI note 60), and C5 is the note C one octave higher (MIDI note 72). If the octave is not specified, it defaults to the octave of the preceding note (or octave 4, for the first note in the melody). Finally, the optional underscore indicates that the note is tied to the following note.

In the user interface, a dropdown box provides a list of named melodies that have been saved on disk. Selecting one of these loads the melody, avoiding the need to retype the long melody representation. Additionally, a separate utility program was developed to convert back and forth between the standard MusicXML format and Musicat’s text format, greatly facilitating music entry by allowing use of a separate notation program such as Finale or Sibelius.

Once a melody has been entered or selected, clicking the “Run” button starts the “listening” process; this is the main loop of the program, described later in this chapter. This can be paused or stopped with buttons in the user interface. The speed of the run can also be changed with the slider bar in the top right corner of the window. This slider adjusts the duration of a very frequently occurring but also very short delay built into the program to slow things down. Additionally, it affects the frequency with which the screen is redrawn; the fastest speeds are attained with the screen is redrawn less often.

The detail slider at the right edge of the window adjusts how many structures are displayed in the workspace view to the left. A high detail level shows everything, while a low

¹² Standard pianos have only 88 keys, indexed by MIDI note numbers 21–108.

detail level causes only the strongest structures to be displayed. Finally, a checkbox is provided to control whether or not future expectations (shown in purple) are displayed in the workspace.

Development Interface for Musicat

The standard interface shows the program acting on a single melody at a time. It takes at least several seconds to run, depending on the speed selected. The run time is not, as one might expect, primarily due to computation (running codelets). In fact, the bulk of the time is spent in redrawing the user interface many times per second to show the workspace as it changes.

Thus, to speed up development, I wrote a separate interface that does not redraw the screen while Musicat runs. Instead, it performs an entire run and then displays the final result. This is not as informative as watching a run in real time, but it is much faster. In order to speed development even more, I integrated this version of the user interface into my development environment, Visual Studio, as an “add-in” panel that is displayed right next to the code (Figure 2). This might seem like a technical detail, but it was actually extremely helpful in making modifications to the code and seeing the results nearly instantly. I set up the program so that it would re-run two entire melodies and display the

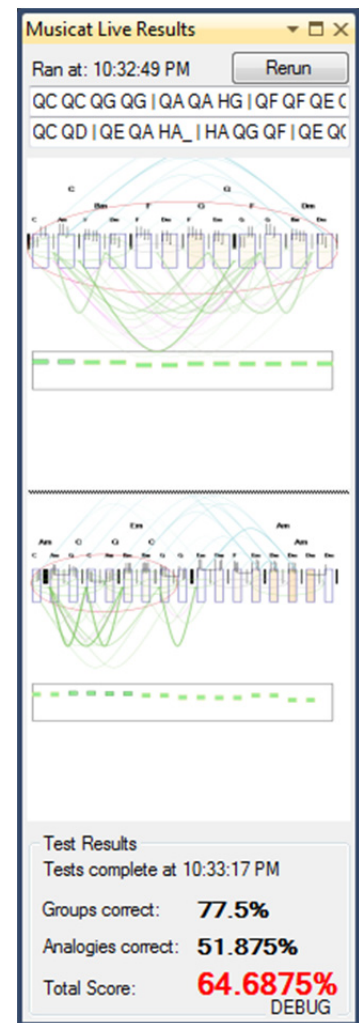


Figure 8.2:
Development interface.

results each time I recompiled any of the code. In Figure 2, the melodies are input as text at the top of the panel; the results are shown, stacked, below the text.

In addition to the graphical instant-run results, I also set up a test suite consisting of a set of melodies with desired resulting group structures and analogies. Each time the code is recompiled, the test suite is rerun: each melody is run through Musicat several times (to account for different results due to randomness), and the percentage of correct groups and analogies is computed across all runs. These percents are displayed at the bottom of the panel and averaged to give a total score (shown in red). The test suite takes a long time to run, but it does so in the background. The total score is a helpful guide in trying out new codelets or changing parameter values; I can make a change, recompile, and see instant results for two particular melodies in terms of the displayed grouping structure and analogies, and then if I wait a while longer I will see the effects of the change on the total score for the test suite.

PROGRAM INITIALIZATION AND ASSUMPTIONS

After a melody has been input by the user, Musicat makes some assumptions about it to simplify the problem. Future versions of Musicat would ideally avoid this phase — see Chapter 8 for discussion. The assumptions have to do with the key and the metric structure of the piece. Key is handled in a very simple way: the program assumes that the melody is in the key of C (either major or minor mode), so all input melodies must be given in C. Musicat does not know about modulation.

Meter is treated in a similarly simplistic manner: Musicat assumes that the entire melody is in the same time signature. This restriction was not present in earlier versions of the program, but the current version considers the measure to be a basic unit of time, so it is important that each measure have the same duration. Importantly, Musicat is restricted to making groups that start and end at a measure boundary, with one type of exception,

described below. Admittedly, this is overly restrictive, as many melodies have phrases starting or ending mid-measure. Additionally, some small-scale structures (such as a simple group of four sixteenth notes) could be considered to be groups; however, Musicat focuses attention at a larger scale for its groups. Again, see Chapter 8 for discussion.

Exceptions to the rule of groups starting and ending at measure boundaries are found in melodies starting with a pickup note (or notes). Pickups are particularly common in 3/4 meter (for example, a 3/4 melody may begin with an unaccented quarter note on beat 3, followed by an accented downbeat), but they can occur in any time signature. Multiple measures can even serve as pickups at a higher metrical level (*e.g.*, in the First Movement of Mozart's 40th Symphony; see Figure 3, where the first strong beat at the hypermeasure level is shown with an accent mark¹³ in the second full measure), but these more complex cases are not handled by Musicat.



Figure 8.3: Hypermetric upbeat in Mozart's 40th Symphony, First Movement.

For melodies with pickup notes, Musicat uses a simple trick: in its internal representation, the bar lines of the input melody are shifted to the left by the duration of the pickup notes, so that the pickup notes seem to Musicat to be the start of a measure. These shifted measures are used when creating group structures. For example, with a one-beat pickup in a 3/4 piece, groups must all start on beat three of a measure and end after beat 2 of another. The original beat positions are preserved, however, for calculations involving things such as beat *strength*. See Figures 4–5 for an example of this shifting process, where metric accents are indicated with accent marks in the second figure, and grouping structures are

¹³ The accent on this downbeat is not literally present in the melody line, but the strength of the beat is implied by several factors, including a low G in the cello and bass. See Bernstein (1976) for a detailed discussion.

indicated with slurs. The internal restriction to forming groups at bar lines in the shifted version of the melody (Figure 5) results in correct grouping structure in the unshifted version (Figure 4).

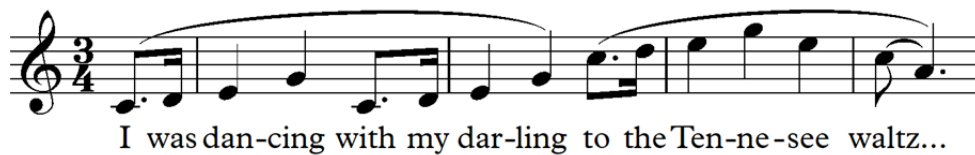


Figure 8.4: Tennessee Waltz melody with pickup beat.



Figure 8.5: Tennessee Waltz melody with shifted bar lines.

MAIN PROGRAM LOOP

Once a melody has been given to the program, Musicat is ready to start simulating the listening process. But how exactly does Musicat work? How does it simulate real-time listening? We can look at the main program loop to get a high-level picture of the process. The program follows these steps until the processing of the melody is complete:

1. Add the next note to the Workspace.
2. Determine a number k of codelets to run for this note. This number is directly proportional to the duration of the current note (measured in sixteenth notes), and the constant proportion used allows for a simulation of different musical tempi.
3. Compute the quality, or happiness, of the representation of each measure in recent history.

4. Fill the Coderack with a mixture of random codelets and codelets selected in a top-down manner to act on any measure whose happiness is low.
5. Allow a small number of codelets to run, and then return to step 2 (unless k codelets have already been run).
6. By now k codelets have been run, so return to step 1 if there are still notes to process.
7. No more new notes remain to process. Musicat runs a set number of codelets (corresponding to a few measures of musical time) to wrap up its processing of the final notes.

Note that in step 2, the selection of k codelets results in simulated real-time processing. If our goal were to perform strict real-time processing, this would be insufficient; moreover, different codelets require different amounts of time to run. We might imagine a version of Musicat that uses asynchronous processing to add notes to the Workspace in real time, depending on the tempo of the music, and unrelated to computation time. However, because Musicat runs large numbers of codelets, we could in principle compute an average computation time (in seconds) per codelet (if we are careful to design codelets such that each one is guaranteed to run quickly and complete in a short time¹⁴), so choosing k proportional to note duration results, on average, in a real run-time that is proportional to the results we might attain with an asynchronous method. Also, for this model real time is inconsequential – if the program ran at half or 10% or 0.01% of real-time speed, that would still be acceptable, because the goal is cognitive modeling, not real-time application.

¹⁴ Codelets are conceptually very small pieces of code. With care we can ensure an upper bound on time complexity of each codelet that is constant with respect to the length of the melody and the number of items in the Workspace.

WORKSPACE

As in other FARG projects, the Workspace represents working memory and the perceptual structures built therein. The use of the Workspace in Musicat is perhaps more similar to that of Seqsee than of any other FARG project because both programs deal with temporal sequences. However, to aid in modeling the flow of time, Musicat's Workspace is not confined to structures presently available in working memory, but instead also contains older structures that have already faded from conscious awareness. A strict interpretation of the term "Workspace" might have required old structures to move into a longer-term memory, but in avoiding this transfer process for individual melodies, Musicat maintains a more blurry notion of the distinction between longer- and shorter- term memory. (A separate long-term memory that would be useful for making analogies between different pieces of music would be a useful addition in a future version of the program). In addition, the Workspace contains structures temporally located not only in the present and past, but also in the future: it may contain "hallucinated" structures that it expects to occur later.

Objects in the Workspace belong to roughly three categories: raw inputs, generated structures, and future expectations. How do objects get created in the Workspace? At the start of each run of the program, the Workspace is empty. The first action the program takes (Step 1 in the main program loop above) is to add the first note of the melody to the workspace. Then the Coderack is allowed to start processing by running codelets. Additional notes are added as time passes (as described above). Codelets add structures, destroy structures, analyze structures, and annotate the Workspace to guide other codelets; almost all of the objects in the Workspace, aside from the notes and measures of the melody, are generated by codelets.

The first category of objects — raw inputs — is made up of the unaltered notes added to the Workspace by the main loop of the program. These notes are held fixed in the

Workspace; they can't be modified or deleted. The second category comprises the bulk of the stuff of the Workspace: these are structures built by codelets, such as groups of notes or analogies between groups. All these structures are "hallucinated" by the program. They aren't as fixed as the raw input notes, but they can gain strength and become just as significant (or even more so). Finally, the third category is composed of structures such as groups or notes that are expected to occur in the future. These are also hallucinated structures, but they are much less tangible than other structures because they might never become "real" structures in the workspace. These expectations — either for certain groups to form, or for certain notes to occur — might be fulfilled or denied, depending on what comes next in the input stream.

Some additional objects in the Workspace are given to the program "for free" to reduce the complexity of the listening problem. These include the key, time signature, and measure-level metric structure of the piece. For example, all melodies are presented in C major or A minor, and the program is told which key is in use. Similarly, the time signature is given up-front, and it is assumed to be constant throughout a melody. Likewise bar lines are implicit: the program is given the metric position of the first note (and implicitly all successive notes), so it doesn't have to decide whether to interpret the first note as a downbeat in 4/4, or an upbeat in 3/4 time, for instance.

CODERACK

The Coderack is a critical component of the architecture, but is also the most straightforward one to understand. Just as in other FARG projects, the Coderack provides simulated parallel processing of codelets in much the same way as a modern computer operating system simulates multitasking of processes and threads. It stores a set of codelets to be run later. Each codelet is assigned an urgency by its creator, which is either another codelet or the main program loop. Only one codelet runs at a time, but because codelets

perform small units of computation, many codelets run in a short span of time. At each iteration of the main program loop, the Coderack selects the next codelet to run. Higher-urgency codelets are more likely to be run next than lower-urgency codelets. Urgency is important for two reasons: it suggests a rough relative ordering of codelet execution, and it helps determine which codelets will eventually execute and which might be deleted without ever having run. Many codelets are posted to the Coderack, but not all posted codelets get a chance to run (since only a fixed amount of computation time is available). Periodically, the Coderack is cleaned up by removing the oldest codelets — that is, codelets that have been waiting on the Coderack for the longest time — one at a time until the number of waiting codelets is acceptably small (50 codelets, for example). Only 10–20% of codelets typically are ever actually run; the others are eventually purged in this cleanup process.

Choosing the next codelet to run is the central step in the description above. The basic algorithm for making this choice is the well-known “roulette-wheel” selection method, commonly used, for example, in genetic algorithms to decide which individuals will survive into the next generation (based on their fitness scores). Each codelet’s urgency value is divided by the total urgency of all waiting codelets to determine the size of the “pie” allotted to that codelet in a random virtual spin of the roulette wheel.

Previous FARG programs have used two different modifications to the urgency value before it is used in codelet selection. In the current version of Musicat these modifications were not used, but it is important to understand why. The first modification was one used in the Tabletop program: the urgency is reduced if the codelet is a child of another codelet (as opposed to a child of the main program loop itself). Urgency is reduced for each generation. This strategy was used in Tabletop to prevent an “urgency explosion” problem, detailed by French (1992). I also implemented this strategy in Musicat, but I found it to cause worse performance, so eventually I removed it. I believe that in Musicat, codelets do not spawn

other codelets as frequently as in Tabletop, and when they do, they are generally very useful. Additionally, the structure of the codelet-creation graph in Musicat is virtually loop-free (see Figure 6); the only loops in the graph are self-references, where a codelet may create child codelets with more specific parameters. Arrows in the figure represent parent codelets creating child codelets; for example, the **Suggest Parallel Analogy** codelet may generate **Create Analogy** codelets, which may in turn generate **Meta Grouper** or **Look for Contour Relationship** codelets. These last two types of codelets, however, do not themselves generate any more codelets directly. From the graph, we see that the longest chain of codelets (allowing only one self-loop, which is reasonable), has length four. At present only 15 codelet types can create other codelets; the other 25 codelet types just run and terminate without generating any children.

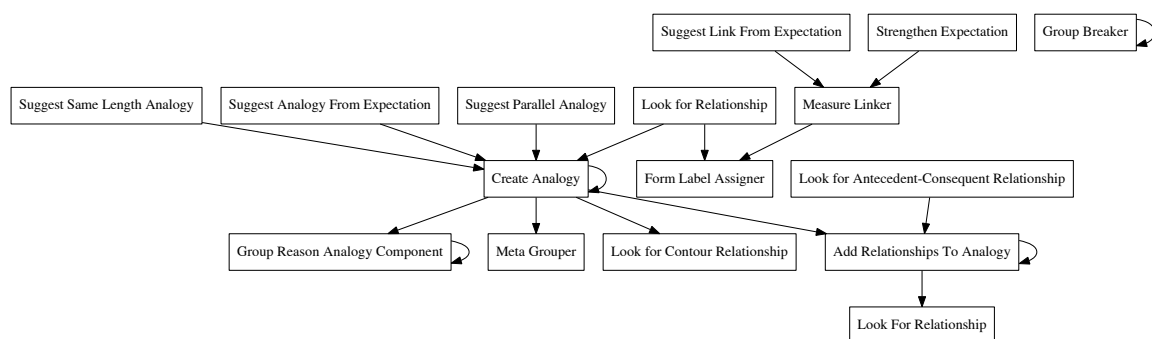


Figure 8.6: Codelet-creation graph. Boxes at the top of the graph represent codelets that can generate codelets of the types below, shown by arrows.

The second potential modification to a codelet's urgency has to do with the current temperature of the Workspace: in other FARG programs, the distribution of codelet urgency values is modified to be flatter as temperature increases. That is, urgency plays less of a role as the temperature increases, which makes the codelet selection process more random. As temperature decreases, the urgency values are used more and more in the normal manner, and processing becomes less random. Note that in contrast to many other FARG programs,

Musicat has no notion of global temperature, although it does have local temperature based on the strength of structures associated with each measure. Specifically, we can compute the happiness H of each measure and derive its temperature $T = 100 - H$. Thus the lack of global temperature doesn't actually pose a major problem; instead of global temperature, Musicat could use the average temperature of the measures of music in the recent time window (4 or 8 measures, for instance). If the recent measures have high average temperature, the Coderack could be made to act more randomly, ignoring urgency values. However, if we know that certain measures have high temperatures, and hence lower happiness values, simply using this temperature value to increase randomness in codelet selection would not be sufficient to drive codelet activity to the individual measures needing attention. Instead, Musicat uses the more focused mechanism of adding to the Coderack additional high-urgency codelets that act on measures with low happiness. Temperature values are still used in stochastic decisions made by codelets, such as deciding which of two competing groups wins a "fight", but for codelet selection, local high-temperature regions are used to indicate places to assign more computational effort. Urgency values are especially crucial when temperature is high precisely because urgency is the mechanism by which the main program loop tells the Coderack which codelets are the most important to run next, so Musicat does not weaken the effect of urgency as temperature increases. In other FARG programs, urgency-flattening due to high temperature made sense because temperature was computed globally; in Musicat, temperature is computed locally and thus has only a local effect.

CODELETS

The Workspace provides a place for Musicat to build a representation of a melody, and the Coderack provides a mechanism for storing future codelets to run and for choosing

what to do next, but the codelets themselves, of course, do the real work of Musicat, building up perceptual structures in the Workspace. Just as in other FARG models, each individual codelet performs a very simple, limited task (such as noticing that two particular measures have the same rhythmic pattern), and then it dies. Any codelet's action is very simple. Codelets often work at cross-purposes, performing conflicting actions in the Workspace. Paradoxically, it is precisely this frenzied struggle involving competing goals that results in fluid perception; the seeming chaos of a myriad of myopic single-minded codelets all performing their actions independently of each other provides the necessary substrate for emergent creative behavior. Ideally, the collective effect of many codelets operating in parallel is simulated perception that responds in a fluid manner to the changing structures in the Workspace.

Codelets may be loosely grouped into two broad categories: those that generally contribute to top-down perception and those that contribute to bottom-up perception, although there is some overlap, which I discuss later. Before describing these categories, I mention some features of codelets in general.

Features Common to All Codelets

Designing codelets is quite unlike the design of typical software. Several unique circumstances come up in their implementation, stemming from the stochastic nature of the program (codelets may run in any order). Implementing codelets thus has a lot in common with writing code capable of running in a multitasking computer environment. Codelets must be able to interact with a Workspace that is shared by all codelets in the system. Additionally, musical time is passing — each time a codelet completes, another unit of simulated musical time “ticks” — and codelets must focus their attention on structures that were created “recently” in musical time. This requirement of focusing on the present and

recent past is enforced by other code in Musicat, so that individual codelets will not process structures occurring too far in the past.

To facilitate the arbitrary order of execution of codelets, each codelet is responsible for performing a series of validity tests at the start of its run. Recall that codelets in Musicat come from one of two sources: either they are created completely at random by the main program loop, or they are created in a more directed, top-down manner. In the first case (random creation), each codelet is created without any parameters specified (such as which measures to examine or which group to break); these parameters are filled in later when the codelet runs. This causes no problems related to arbitrary execution order. In the second case (top-down pressure), however, codelets, when created, are given some specific direction about which Workspace elements to use when they run. These parameters, such as “the group of measures 3 through 6” in an **Extend Group Right** codelet, may no longer be valid by the time the codelet runs. In this case, the group in question existed when the codelet was created but might have been destroyed by the time the codelet is picked to run. In this case, the solution is simply to make sure the group still exists before the codelet runs, and if not, to perhaps try to find a different group to operate on. In general, each codelet must evaluate a set of preconditions to verify that its set of parameters is still valid before it is given the green light to run. For codelets with many parameters, this can be surprisingly tricky (just as ensuring thread-safety of multithreaded code is tricky). However, as long as it is done carefully for each codelet, arbitrary run order works without any problems. Because each codelet runs atomically (codelets themselves do not run in parallel in this version of Musicat), these precondition checks also ensure the integrity of the Workspace, even though each codelet can change the contents of the Workspace. Note that it would be possible to implement Musicat to run multiple codelets in true parallel (*i.e.*, on multiple threads) but even more synchronization constructs would be required; because of the highly dynamic

nature of the Workspace, this might not result in much performance improvement without a great deal of careful planning.

In order to enforce the idea that codelets must focus their attention on recent perceptual structures that still exist in working memory, codelets are expected to interact with the Workspace through a set of intermediate C# functions that can restrict their access as necessary. In terms of software design, each object in the Workspace (including the whole Workspace itself) is represented by a C# class that provides methods for codelets to interact with the objects. These methods take care of preventing access to objects that are too far in the past, and they do so in a consistent way for each type of codelet. As an example, imagine that a codelet tries to create a new group. The codelet may detect groups that would conflict with the proposed new group, so it first enters each conflicting group in a probabilistic competition with the proposed group, and removes the conflicting groups if they all lose their fights. The codelet would call a standard Workspace class method to remove the losing groups, but this method has added checks built in that stop the removal if other conditions are not fulfilled. For example, if the group to be removed occurs too far in the musical past, it will never be removed. Overall, a great deal of code for reading from and writing to the Workspace is encapsulated in Workspace-related objects.

Top-Down Codelets

The terms “top-down” and “bottom-up” can be slightly confusing. In each field that uses these terms, these terms tend to be used differently. Even within a single research group, such as FARG, the terms can be a bit ambiguous. For example, in his dissertation on Phaeaco, Harry Foundalis writes:

There are bottom-up and top-down codelets. The former act directly on structures in the Workspace without any prior information about what

should exist there. Top-down codelets, in contrast, carry out actions on Workspace structures with an eye to creating specific types of higher-level structures. (p. 132)

I use the terms slightly differently, although in a similar spirit. Specific codelets that have been posted to the Coderack in Musicat may be called “top-down” or “bottom-up”, but codelet *types* generally do not have an intrinsic top-downness or bottom-upness to them. In Musicat, “top-down” instead refers to those codelets that have been generated as a result of context-based pressures, which I call “top-down” pressures. The goal of these pressures is generally to create higher-level structures, but the individual codelets generated may not necessarily have these specific goals.

For example, if Musicat has perceived an analogy between measures 1–2 and measures 5–6 in a melody, a **Suggest Parallel Analogy** codelet might be generated, with this known analogy as a parameter. The codelet might then look for an analogy between measures 3–4 and 7–8. But if there is a conflict, such as a group spanning measures 3–6, a **Group Breaker** codelet might be added to the Coderack. This codelet would still have a top-down genesis, although the more typical use of a **Group Breaker** would be to destroy random weak groups, in a manner that I would call “bottom-up”. In general, particular codelets created in response to the big-picture context are called “top-down codelets”, just as the pressures to create such codelets are called “top-down pressures”.

Bottom-Up Codelets

Bottom-up codelets, in contrast to top-down, are not generated in response to contextual pressures from the system. Instead, the Coderack is periodically refreshed with a pool of codelets that randomly choose small-scale elements of the Workspace to analyze, where the random choice is biased towards analyzing the elements that were heard the most recently in musical time. A typical example of a bottom-up codelet is the **Measure Linker**

codelet, which looks for simple rhythmic similarities between measures. If this type of codelet is generated by the main program loop, it selects a pair of recent measures at random to analyze. Most of the time a random pair of measures will not wind up getting linked by such a codelet. Occasionally, however, a useful link is formed, which stays in the Workspace and contributes to larger constructions. But, as discussed above, **Measure Linker** codelets may also be generated by other codelets such as **Suggest Link from Expectation** codelets, which would indicate pairs of measures that the **Measure Linkers** should try to link; in this case, I would call these particular **Measure Linkers** “top-down”, not “bottom-up”.

TEMPERATURE

The notion of temperature was used in Copycat, as well as in several related FARG programs, to balance the need for undirected and chaotic action, when the program was far from a solution and needed to explore more “wild” ideas, with more focused and deliberate action, when a good solution was taking shape. The Workspace in these programs had a global temperature derived from the quality and cohesiveness of the Workspace’s structures. Low-quality structures caused the temperature to go up and thus increased the randomness of codelet actions, while high-quality structures reduced temperature and thus randomness. Additionally, over time, the temperature was made to decrease gradually, as in simulated annealing, helping the program to settle down to some solution (even if of low quality).

Musicat also incorporates temperature, but two changes were necessary for temperature to make sense in the context of real-time listening. These have to do with the simulated-annealing idea and with a need for local (as opposed to global) temperature.

Whereas Copycat’s goal for each program run was to solve a single analogy problem, with the complete Workspace structure constantly available in memory, Musicat creates many different structures and analogies in one run, and older structures fade into the past as

new notes are “heard”. Copycat’s Workspace can be represented naturally by a diagram of fixed size in 2-D space (*e.g.*, as a rectangle with all the pieces of the analogy contained within), but Muscat’s Workspace is better visualized as a 2-D diagram where time has been mapped onto the x -axis and objects are flowing to the left as they gradually move from the present into the past. Because time is moving along naturally, the enforced gradual decay of temperature doesn’t make at much sense here. Musicat generates structures to make sense of recently-heard notes, but as new notes of the melody are perceived, old notes and structures stabilize as they leave the working-memory area, without the need for a simulated-annealing process.

The other difference in temperature between Copycat and Musicat is more significant. Using a single temperature value in Copycat is reasonable because its goal is to create one unified analogical mapping explaining the input and suggesting a solution. Why can’t a similar global temperature be defined for Musicat? It is tempting to think that Musicat’s goal is to create a single coherent musical parsing of a melody (such as the Time-Span Reduction in GTTM), and a temperature value could be computed that would describe the coherence of the generated structure. However, this picture is too perfect to be a good model of the human listening experience. Instead, some portions of a melody may indeed be heard as part of a strong structure, but other parts may be less understood, and the listener will simply move on and continue listening to newer notes without ever coming to a complete understanding of that part of the melody. Thus, Musicat computes temperature over small ranges of music; there is no global temperature.

Although there is no global temperature, the window of recently-heard music has special significance and we can make a rough analogy (for program-architecture purposes) between the music currently active in working memory and the entire Copycat Workspace, which also represents structures in working memory. Thus, in Musicat, the temperature of

the range of measures in working memory (say, eight measures) is used to modify certain probabilistic codelet actions. As was explained earlier, urgency values are not modified by this temperature value. However, stochastic decisions such as determining which of two structures will win a “fight” are influenced by this temperature.

In Musicat, it is often easier to think of a “happiness” value instead of temperature. Recall that happiness and temperature are related by the equation $T = 100 - H$, where both H and T range from 0–100. For example, part of the main program loop computes the happiness of single measures in the recent history of the Workspace and creates codelets to act specifically on structures involving those measures with low happiness.

SLIDING TIME WINDOW OF PERCEPTION

As was mentioned above, Musicat focuses its attention on the most recent measures of a melody and the most recent structures in the Workspace. New structures come and go rapidly as codelets try out different ideas, but as time passes, Musicat’s perceptual structures become more fixed. Eventually, structures that are old enough become unchangeable. This reflects our intuition that listeners can retroactively change their perception of musical structures but only for a short time. Presumably, this time span is determined by the length of time that structures persist in working memory, although I’m not aware of any research specifically on the subject of retroactive listening and working memory.

In Musicat, two distinct methods are used to make structures stabilize as time passes. First, groups that lie several measures in the past can receive a strength bonus based on how distant they are from the most recently perceived note. For example, imagine a piece where measures 1 and 2 are identical to measures 3 and 4. Also imagine that Musicat has generated a group G1 containing the first two measures, and a second group G2 containing measures 3 and 4. If measure 4 has just been completely presented to the program and measure 5 is

starting, G1 and G2 might look identical, but G1 would likely have a higher score because it is further in the past. Note that this bonus is based solely on the temporal distance between the end of the group and the present moment; the age of the group, in terms of how long it has persisted in the Workspace after its creation, is not a factor here. Once enough musical time passes (4 or 8 measures, based on a constant parameter set in the code and increased slightly for large groups), any groups that exist are considered to be permanent, and are immune to further changes. This simulates the idea that eventually any groups constructed in the Workspace will stop being part of active working memory and exist only in a longer-term storage. This also helps ensure that Musicat's algorithms have a bounded time complexity, and makes real-time processing computationally feasible for melodies of unlimited length.

A second thing that helps older Workspace objects stabilize is the manner in which codelets decide which measures, groups, or analogies they should act on. Codelets choose objects to work on simply by asking the Workspace object to provide a recent object of the desired type at random. For example, a codelet may make a request such as "Find a recent measure". The Workspace will choose one probabilistically, according to a decay function that favors the most recent objects. In this way, most codelets act on very recent objects in the Workspace, but sometimes they will act on objects further in the past (as long as those objects have not become permanent).

One unresolved issue in the program is how to best deal with larger-scale structures in the Workspace. Listening happens in a mostly-linear fashion as time progresses, but in addition to a certain amount of retroactive listening that occurs as we reprocess music stored in working memory, we are also listening in a hierarchical way. It is a subtle blend of linear and hierarchical pressures that leads to hearing music as having a particular grouping. How do listeners (and how should Musicat) balance these two pressures? Sometimes a local linear structure (*e.g.*, a scale fragment) sounds like a group, but in other cases the same structure

might be broken into two groups as a result of higher-level (hierarchical) grouping pressures. Fortunately we get some balance for free: time keeps moving, forcing decisions to be made quickly. But this also causes other difficulties: what happens when large structures have moved too far into the past?

Once a listener has perceived large objects such as phrases, periods, or entire sections as “chunks” in memory, it is possible to reason about these larger structures. Musicat can make long-distance analogies between chunks, but the creation of large-scale groups is difficult when some of the elements of the group are no longer in Musicat’s short-term window of perception. Perhaps a mechanism for temporarily reactivating older groups would solve the problem. In Musicat, the Workspace represents both short-term working memory and longer-term memory (for structures which are far in the past and have achieved “permanent” status); this blending-together of short- and long-term memory is architecturally appealing, but a separate medium- or long-term memory space might help clarify some of the issues involved in listening to long melodies with large-scale structures.

Objects in the Workspace

The primary output from a run of Musicat is a representation of the “heard” melody. This representation is simply the contents of the Workspace at the end of a run (although the dynamic, changing representation as time flows during a run is of great interest as well). This section describes the various objects that can be found in the Workspace.

NOTES

Notes are simple: as described above in the section on Musicat’s user interface, they are made up of a symbolic representation of pitch such as “60”, which stands for the pitch of Middle C, as well as rhythmic information. (Note that individual codelets may hear a note as

“the tonic in C major” or “raised third degree of an A \flat minor chord”, but in this version of the program — in contrast to earlier ones — these interpretations are created temporarily by codelets and not persisted as part of the Workspace structure.) The note’s start and end time are given in units of sixteenth notes, relative to the beginning of the measure which contains it. Notes may also have a tie to the following note. When a note is tied, Musicat understands that the note is not rearticulated, so that operations requiring a list of attack points, for instance, ignore notes where the previous note was followed by a tie. Note that ties cause some subtle issues that must be handled carefully in the software. For instance, when shifting measures internally to account for pickup beats, sometimes ties will need to be created to account for long notes which cross a bar line after the shift operation.

MEASURES

Each note in the Workspace must be contained within a measure. A measure stores a list of its notes and a time signature. Optionally, a measure may store an associated pitch alphabet. Musicat’s measures are the smallest elements which can be members of a group. Although codelets can examine the notes contained in a measure, in many ways the measure is the atomic musical unit in Musicat.

RHYTHM-BASED MEASURE LINKS

Rhythmic similarity plays a key role in Musicat. It is the most basic kind of similarity and forms the basis for more complex types of relationships between musical structures. Thus, there is a special object in the Workspace that indicates rhythm-based similarity, simply called a *measure link* internally, as opposed to the more descriptive “rhythmic measure link”, because rhythmic similarity is the natural, default type of similarity Musicat notices. In other words, the “rhythmic” part is implied. (Other, more general links, which may involve pitch

information, are called *relationships*). When two measures are found to be (rhythmically) similar to each other, a measure link may be created (by codelets) to represent the discovered similarity.

Measure links appear in the user interface as arcs above the staff. Like most objects in the Workspace, each measure link has an associated strength value in the range 0–100. For a measure link, this describes the amount of similarity discovered between the two measures in question. In addition, the strength can decay over time if the link is not used as a component of any larger structure. Thus, irrelevant links eventually fade away.

RELATIONSHIPS

“Relationship” objects in the Workspace (henceforth referred to as “relationships”) are used to represent all types of similarity between pairs of objects in the Workspace. Each relationship has a strength value (just like measure links do). In addition, each relationship has a specific type. Relationships can be formed as a result of perceived similarity in melodic contour or metric position, or of the presence of a rhythmic measure link, or of two groups having the same initial notes, or for many other reasons.

Although any relationship is *conceptually* an analogy, in Musicat, there is a special *analogy* datatype reserved for describing a mapping between two large structures and their components (read more about the analogy datatype in the “Analogies” section below). For the program, then, relationships and analogies look different — they have different datatypes. To allow codelets that act on relationships to also act on analogies using the same code, I have adopted the following solution: when an analogy is created in the Workspace, a special type of relationship — *an analogy relationship* — is automatically formed in the workspace, in *parallel* with the analogy datatype. An analogy relationship simply links two large objects and indicates that they are similar, so that other codelets can be aware of the

relationship without unpacking all the details involved in the large data structure that describes an analogy. (The *analogy relationship* can be thought of as a “hack” that makes it simple for codelets to act on relationships without understanding the inner details of the relationships.) In general, a relationship represents a cognitive association between two objects, where the details and the reason for the association are not immediately accessible.

BAR LINES

In Musicat, a bar line is more than just the boundary point between two adjacent measures. Bar lines are represented explicitly in the Workspace because each bar line can have a perceived “thickness” assigned to it by codelets, which is modifiable over time.



Figure 8.7: A typical sequence of bar line “thicknesses”.

A bar line’s thickness indicates how strongly an implied breakpoint might be heard between two adjacent measures based solely on their positions in the melody’s metric hierarchy. Equivalently, we can say that bar-line thickness corresponds to the strength of the metric stress of the measure following the bar line. For example, after the second measure of a melody with a straightforward duple hypermeter, we would hear the bar line as slightly “thicker” than the bar line after the first measure. Then the bar line after measure four would be thicker yet, and the one after measure eight would be the thickest bar line encountered up to that point in the melody. The thicker a bar line is, the less likely it is that a group will be formed containing measures on both sides of it (unless the group is quite large itself). Other

representations of hypermetric structure in the literature have used a metaphor of “height” or “strength” attached to the first beat of a measure, but we feel that bar-line thickness more clearly communicates the relationship between hypermeter and grouping boundaries. (Of course, as Rothstein points out (1989), grouping and hypermeter structure may differ by high-level (hypermetric) upbeat, such as shown in Figure 8.3, but Musicat has a strong preference for these two structures to line up exactly.)

ALPHABETS

Pitch alphabets (or simply “alphabets”) were discussed earlier in the context of Larson’s Seek Well program. They are used in Musicat to describe the implied harmonic context of a measure or group. An alphabet is simply an ordered collection of pitches. In Musicat, the pitches of an alphabet repeat in every octave (*e.g.*, a major chord with an added ninth is equivalent to a major chord with an added second). Musicat’s alphabets differ slightly in implementation from those used in Seek Well, in which each regular alphabet was paired with a goal alphabet (a subset of the regular alphabet). In Musicat, each pitch in an alphabet has an additional binary flag marking the pitch as stable or unstable. Stable pitches correspond to pitches in a goal alphabet in Seek Well.

GROUPS AND META-GROUPS

Simple Groups

The simplest kind of group in Musicat is a collection of adjacent measures. One-measure groups are possible in certain circumstances, although they occur quite rarely and are expected to grow to incorporate additional measures. Groups represent musical structures, such as phrases, that are heard as a unit. Each group maintains a list of “reasons”

that it exists (*e.g.*, starting after a thick barline and ending on a long and stable note), as well as a list of “penalties” describing problems that afflict it (*e.g.*, spanning a thick bar line). These lists are created and modified over time by codelets. A group’s strength is calculated based on these reasons and penalties (details will be given in the “Calculating Structure Strengths” section). Groups also maintain a current choice of pitch alphabet, assigned by and modifiable by codelets, which is used to aid understanding of notes of the group in a harmonic context.

Meta-groups

Groups can contain other groups. For example, if a two-measure group G1 spans measures 1–2 and another two-measure group G2 spans measures 3–4, then a group M can be created that contains both G1 and G2. I sometimes call a group that contains other groups a “meta-group”, but for simplicity I use the term “group” for all kinds of groups, whether they contain groups or measures (or both). Elements such as measures or groups that are contained in a meta-group are called *children* of the meta-group; conversely, the meta-group is called the *parent* group of those elements.

Restrictions on Groups

The elements contained in a group must be temporally adjacent. For example, if we continue the previous example by adding another two-measure group G3 spanning measures 5–6, Musicat can make a group M2 that contains all three groups G1, G2, and G3, but it is not allowed to make a group M3 containing only groups G1 and G3.

Groups may contain other groups, but no groups may overlap temporally. Equivalently, we can say that each measure is the direct child of at most one group (measures may not have any parent group at all, but Musicat has internal pressures that push it to try to avoid this situation.) For example, given the group G1 above, it is not possible to create a

group G4 that spans measures 2–3; the two groups would be in contention because they both contain measure 2. Such conflicts between rival groups occur extremely often during each run of Musicat and codelets are considering alternate grouping structures, creating rival groups temporarily in memory for consideration, but for all groups stored in the Workspace the non-overlapping rule applies strictly. Note that requiring non-overlapping groups is a significant restriction in the music domain, and allowing a small amount of overlap would be preferable in a future version of the program. Lerdahl and Jackendoff (1983) demonstrate in GTTM that a single note might simultaneously fill the dual roles of being the final note of one group and the first note of a second group. These two groups, then, would overlap by one note. In spite of this possibility for single-note group overlap, GTTM has a rule requiring non-overlapping groups (Grouping Well-Formedness Rule 4), but it is justified by a discussion of group elision, in which any passage including an overlap of the sort discussed here is seen as a syntactically-transformed version of an underlying musical structure in which the first group ends and then the note is repeated to start the second group. This idea is related to Chomsky's notion of the surface form of a sentence deriving, via syntactic transformations, from an underlying deep structure. In the current version of Musicat, groups must start and end at measure boundaries, so the sort of overlap just described in GTTM cannot occur. For styles of music where group overlap is natural and common, this may present a problem, but for the time being, requiring non-overlapping groups makes sense because groups in Musicat must occur at measure boundaries and in a relatively simple melody, it is unlikely that an entire measure would serve as both the start measure of one group and the end measure of another group.

Sequences

Analogy structures in the Workspace (see below) involve a mapping between two objects, along with their subcomponents. The musical notion of a sequence, on the other hand, often involves three or more copies of a musical pattern appearing successively (in *sequence*, of course). A sequence is thus a distinct object from an analogy for Musicat. Sequences also have a strict definition here: sequences require nearly-exact copies of a melodic pattern, with only minor tonal variations allowed to due transposition. For example, the pattern in the first measure might be repeated in the next measure, but with each note shifted one tone lower in an alphabet such as the major scale. If the same pattern were to continue for a third measure, Musicat might detect this as a sequence. See Figure 8.8 for an example. To be sure, for a human listener, hearing such a pattern as a sequence is indeed analogy-making in action, but treating the notion of sequence as a special circumstance with rather rigid rules was helpful in the implementation of Musicat. This definition of sequence handles many interesting cases, because many sequential patterns in music are quite strict and require only minimal amounts of flexible analogy-making to understand.

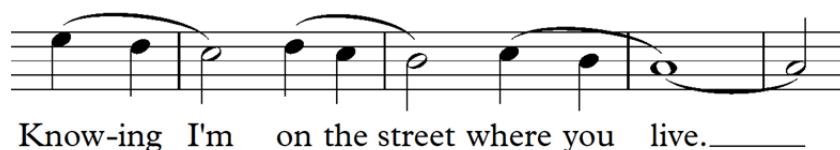


Figure 8.8: A typical sequence.

In Figure 8.8, the three-note pitch pattern (step down, step down) repeats perfectly three times, with each instance starting one step lower. However, there is a very small difference in the third repetition of the quarter-quarter-half rhythmic pattern: the final note is a tied whole note, lasting six beats instead of the expected two beats. Musicat allows the final repetition of a sequence to be different after the initial notes are played as expected. This

allows changes at the end of a sequence, as in Figure 8.8, to be considered as part of the sequence. In the case of threefold repetitions like this one, a change at the end is often necessary to round out a phrase, which likely has a length (in measures) of four, eight, or some other power of two.

ANALOGIES

As was discussed in Chapter 4, analogy-making plays a central and omnipresent role in music listening (not to mention the rest of cognition). In implementing a program such as Musicat, there is a danger of not coming anywhere close to the flexibility and complexity of human analogies, in that one needs to design highly-precise data structures in order for a computer to represent an analogy. Of course, the whole architecture of Musicat and of other FARG programs was designed to support fluid analogy-making; nevertheless, I am reminded of the vast oversimplifications necessary in my program whenever I open up the “*Analogy.cs*” file and see the scant 477 lines of code that define the *Analogy* class in the C# language. With that disclaimer in place, I will describe how Musicat represents analogies. But first, I will explain some small-scale linking structures in Musicat that are not part of the official “*Analogy*” class, even though I think of them as analogies.

At a low level, Musicat represents similarity between objects in the Workspace through rhythm-based measure links and a set of more general relationship objects¹⁵ described above. One might wonder why these types of links are necessary, instead expecting the *Analogy* class to handle all types of connections between objects. Indeed, it would be possible to rewrite the code to make an all-encompassing *Analogy* class, but there are two reasons I avoided doing so. First, as a practical matter, an analogy is an inherently recursive

¹⁵ In fact, the specialized measure links are transformed into relationship objects when necessary, so it is sufficient to talk about relationships alone for the rest of this section.

data structure: an analogy between two groups might be composed of smaller-scale analogies between children of those groups. Any recursive data structure eventually “bottoms out” in structures with no children of their own; for example, think of the leaf nodes of a binary tree. The “leaf nodes” in Musicat’s analogies are precisely the low-level relationship links. Second, because relationships are so low-level, we can think of them as operating closer to the periphery of the nervous system than higher-level analogy-making. Certain types of auditory processing (such as resolving fundamental frequencies from a sound wave, or, perhaps, detecting repetitive rhythmic patterns) happen much more immediately in the brain than other, higher-level processing. Thus it struck me as reasonable that low-level relationships be detected with specialized code, while having higher-level relationships be found using a more general analogy mechanism.

An analogy in Musicat is essentially a mapping between a group on the left (I abbreviate this as “LHS”, for “left-hand side”) and a group on the right (RHS). Left and right are meant with respect to a musical score, so the LHS is a group further in the past than the RHS. The mapping is flexible: not all elements are required to be mapped from one side to the other; in mathematical terms, an analogy is not an isomorphism. It is simply a set of relationships, where half of each relationship corresponds to an element from the LHS of the analogy and the other half corresponds to an element from the RHS. Analogies at a more abstract level (I call them meta-analogies) are beyond the scope of this program. For example, Musicat can map the first four measures of “Sur le pont d’Avignon” onto the last four measures of the melody: the measures on the left are mapped to the measures on the right. But if Musicat were to notice that measure 1 is related to measure 2 by transposition, and furthermore were to notice that measure 3 is related to measure 4 by transposition, it could not use this knowledge to make a meta-analogy between (1–2) and (3–4). Another

restriction on analogy-making is that all analogies are within a single melody, not between different melodies.

An analogy, just like groups, links, and so forth, has an associated strength value. One important difference between the method of computing the strength of an analogy, as opposed to that of a group, however, is that analogy strength is computed using a fixed list of criteria. An analogy's strength can change over time, however, as new relationships are added to the analogy. An incomplete analogy can be formed with a relatively low strength, and the birth of the analogy will trigger a search for completing the mapping, which will increase the analogy's strength, if the search is successful. Analogy strength is discussed in more detail in a following section.

EXPECTATIONS

There are codelets in Musicat that can create groups, analogies, and measure links that are expected to occur in the future. Expectations are formed in a simple-minded manner: when a strong and relatively large structure has been formed that ends at a thin bar line, Musicat expects that structure to repeat. This is admittedly a simple-minded rule of thumb, but it reflects the intuition that musical patterns tend to repeat. Indeed, we can even think of this as an interpretation of "inertia" in Larson's theory of musical forces: a large structure sets up a trivial expectation that another structure just like it is on the way.

When an expectation is formed, it is used to focus the attention of future codelets. For example, if a group that includes measures 9–10 is expected to form, then once measure 9 is heard, codelets will be created to add measure 9 to a new group, and once measure 10 arrives, the codelets will try, with high urgency, to group measures 9 and 10 together.

Musicat's notion of expectations may seem surprising at first, because typical work on expectation is centered on expectation in the pitch and rhythm domains. These are certainly

important and are a high priority for future work on Musicat. In this version of the program, however, consistent with its focus on understanding the higher-level structure of melodies, expectations involve grouping and analogy, not the pitches and rhythms of individual notes.

Calculating Structure Strengths

The Workspace is filled with the perceptual structures listed above, such as groups, relationships between measures or groups, analogies, expectations, and so forth. Each such structure has an associated strength that indicates how strongly committed Musicat is to the structure. These strengths (equivalently, “scores”) range from 0 (extremely weak) to 100 (extremely strong).

The computation of structure strengths is central to how Musicat functions, and much of Musicat’s idiosyncratic listening style emerges as a consequence of which types of structures tend to be scored highly. In the current implementation, we have tried to make Musicat favor the kinds of structures and patterns that are important in Western folk melody. Many of these, however, such as the primacy of rhythmic repetition, or gestalt grouping principles, are rather general and should apply to most musical cultures. Others, such as the tonal functions of the tonic and dominant scale degrees or the notion of melodic sequence, may be more culturally specific.

A common suggestion I’ve received for Musicat is that the scoring functions should be optimized for a particular musical corpus using machine-learning techniques, perhaps to simulate the effect of musical enculturation on a listener. While this is an interesting and tempting idea, it misses the basic point of this work, which is to model fundamental mechanisms of listening (in the general framework of analogy-making). Additionally, the overall architecture of the model, its types of codelets, and the types of perceptual structures

it can make are probably more critical to its listening style than are the details of how it computes scores.

An analogy with computer chess programs may be useful to motivate the philosophy behind our design choices: the best chess programs use brute-force search as their main “trick” to achieve super-human performance. Deep Blue, IBM’s computer that famously defeated world champion Garry Kasparov in 1997, relied on an optimized brute-force search capable of examining 200 million positions per second. In this type of search, the desirability of each board position in the game tree is computed with an evaluation function that assigns a score such as +0.8, indicating that the first player is ahead by nearly a pawn, or -8.5, indicating that the first player has lost nearly the equivalent of a queen. Interestingly, Deep Blue used a quite simple evaluation function that was optimized for speed, not for subtlety of positional understanding. It turns out that the brute-force search algorithm (combined with the hardware’s raw speed) was the driving factor behind the computer’s success at chess; I suspect that practically any quick-and-dirty evaluation function taking into account the value of the pieces on the board for each player would result in a similar level of chess-playing. Kasparov was so stunned by the subtle-looking nature of the computer’s 37th move (Bishop moves to square “e4”), in game 2 of the 1997 match, that he later accused IBM of cheating.

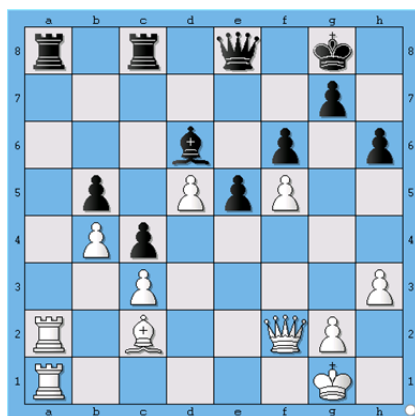


Figure 8.9: White (Deep Blue) to move, just before move 37 (Be4).
Game 2, Kasparov versus Deep Blue, 1997.

These days, however, most strong chess programs (including free programs that I can download and run on my laptop) will, after only a few seconds of computation, come up with the same bishop move as Deep Blue did, which strongly suggests that brute-force search using a wide variety of evaluation functions will lead to this move. Of course, variations in evaluation functions to score positional elements (such as king safety) more highly than aggressive piece placement would naturally result in more defensive play, although even a version of Deep Blue using a more defensive evaluation function would still be capable of brilliant but reckless-looking attacks whenever brute-force search could prove the attack's soundness. Similarly, although the evaluation functions that compute structure strengths are important in determining Musicat's listening style, the overall effectiveness of Musicat as a music listener may be more determined by the program's architecture than by the details of its scoring algorithms. (To be clear, Musicat does not use brute-force search like Deep Blue — Musicat *avoids* brute force! Both programs, however, do make use of evaluation functions.)

The following sections go into more detail about how various structures are scored, starting with simple rhythmic similarity between measures and proceeding through relationship, group, sequence, analogy, and expectation scoring.

SCORING RHYTHMIC MEASURE LINKS

A rhythmic measure link can be formed between two measures that are rhythmically similar. The strength of the link is based simply on the number of note attack-points (articulations) that the two measures have in common. Specifically, the score of the link between two measures m_1 and m_2 is inversely related to the size of the symmetric difference between the two measures' attack point sets, normalized by the total number of attack points.

$$\text{symDiff}(m_1, m_2) = \# \text{missing attacks in } m_1 + \# \text{missing attacks in } m_2$$

$$\text{score}(m_1, m_2) = 100 \times \left(1 - \frac{\text{symDiff}(m_1, m_2)}{\# \text{attacks}_{m_1} + \# \text{attacks}_{m_2}}\right)$$

For example, given the measures (H QQ) and (QQQQ), the attack on beat 2 in the second measure is missing in the first measure, so the symmetric difference is 1. There are three attacks in measure 1 and four in measure 2, so the score is given by

$$100 \times \left(1 - \frac{1}{3+4}\right) = 100 \times \frac{6}{7} \approx 86$$

This is a high score: most of the attacks are present in both measures. (The maximum possible score, achieved only when the two rhythms are identical, is 100.) If we change the second measure to (QQ H), however, then the symmetric difference is 2, the total number of attack points is 6, and the score is approximately 67. In this case, the “correct” part of the score come from both measures having attacks on beats 1 and 3; the missing attacks on beats 2 and 4 in the first and second measures, respectively, reduce the score.

SCORING RELATIONSHIPS

Relationships between measures or groups are the key objects used in forming analogies between groups. Recall that relationships in Musicat are used to represent links that have been found between two structures, not the details of the links. This allows Musicat to easily build more complex analogy objects from simpler relationship objects. The score for a relationship is based on its type, and is calculated by the codelet that creates the relationship in the first place. Details follow for each relationship type.

Types of Relationships

Identical Rhythm, Similar Rhythm

The simplest relationship types — “Identical Rhythm” and “Similar Rhythm” — mirror the primal “rhythmic measure links” in the Workspace. For each such link, an Identical Rhythm relationship is formed if the link had a score of 100; otherwise, a Similar Rhythm relationship is formed.

Start Identical Rhythm, Start Similar Rhythm

Two measures or groups can be assigned a Start Identical Rhythm relationship if they start with identical rhythmic material but diverge later. The “start” of a group or measure is defined as either half the duration or eight beats, whichever is shorter. The score is computed just as for a Rhythmic Measure Link, described above, but using only the start region for the computation.

Analogy

Recall from the “Objects in the Workspace” section above that when an analogy is formed, an Analogy *relationship* is automatically created in the workspace to mirror the

analogy and is used to represent it in higher-level analogies. Unlike other relationship types, an Analogy relationship has a strength that is dynamically computed each time it is used, by recomputing the strength of the source analogy (the analogy object that gave rise to the Analogy relationship). The relationship strength is identical to this computed analogy strength.

Melody Contour

A Melody Contour relationship can be created if two measures or groups have a similar-enough melodic shape. The contour of a selected phrase is represented by a string of symbols representing the difference in pitch between adjacent notes. Five symbols are possible in a contour string (Table 2):

U	Step up (1 or 2 semitones)
d	Step down (1 or 2 semitones)
U	Leap up (> 2 semitones)
D	Leap down (> 2 semitones)
–	Sideways motion (repeated note)

Table 2: Contour symbols.



Figure 8.10: Twinkle, Twinkle.

For example, the first four measures of the “Twinkle, Twinkle, Little Star” melody (CCGG | AAG | FFEE | DDC) (Figure 8.10) would be described by the following string:

–U–u–dd–d–d–d

Observe that for a melody of N notes, there are only $N-1$ transitions between notes; this melody has 14 notes and 13 symbols in its contour string. This becomes more apparent when we consider the same melody broken into a pair of two-measure phrases of 7 notes each (“CCGG | AAG” and “FFEE | DDC”): the second instance of the symbol “d” will not appear because the transition between the two measure-pairs is not represented, so now there are 12 symbols total (Figure 8.11 & Figure 8.12).



Figure 8.11: Measures 1–2.

–U–u–d



Figure 8.12: Measures 3–4.

–d–d–d

To compute the similarity between two melodic contours, such as between “–U–u–d” and “–d–d–d”, Musicat first computes an “edit distance” between strings, where each insertion, deletion, or substitution operation adds 1 unit of distance. The distance between these strings is 2; it takes one substitution to replace the “U” with a “d” and another to replace the “u” with a “d”. Another one-time operation is allowed in edit-distance computation, also with a cost of 1 unit of distance: one contour string can be replaced with its “vertical” inverse, where each “U” or “u” turns into “D” or “d”, respectively, and vice versa. In our example, the string “–U–u–d” could be replaced with “–D–d–u”, and then compared with “–d–d–d”; however, this new string still has distance 2 from the second contour string, for a total distance of 3, so the inverse route isn’t helpful in this case. In contrast, the two melody fragments “CDEFG” and “GFEDC” yield the contours “uuuuu” and “ddddd”, and without the inversion operator they would have distance 4, but allowing this operation results in a distance of 1 between these contours.

Once the edit distance has been computed, the similarity score (which is also the strength of the Contour relationship) is based on the distance divided by the maximum possible distance:

$$\text{ContourSimilarity}(x, y) = 100(1 - \frac{\text{dist}(x, y)}{\text{maxDist}})$$

where

$$\text{maxDist} = \max(\text{length}(x), \text{length}(y))$$

The contour similarity of “CDEFG” and “GFEDC” is thus:

$$100\left(1 - \frac{1}{5}\right) = 80$$

The similarity between the two Twinkle, Twinkle measure groups (“CCGG | AAG” and “FFEE | DDC”) is:

$$100\left(1 - \frac{2}{7}\right) \approx 71$$

Note that this last score is higher than one might expect at first based on the melodies’ shapes; the similarity comes from the three pairs of repeated notes in each group.

Antecedent → Consequent Tonality

Groups may be perceived (by codelets) as ending in either a tonic or a dominant tonal function. For example, a stepwise descent to the tonic might be perceived as having a strong tonic function (implying a I chord), while a relatively long note on the dominant (G) or the supertonic (D) might be perceived as having strong dominant function (implying a V chord). An Antecedent–Consequent Relationship may be generated for a pair of adjacent groups of which the first has been perceived as dominant and the second as tonic. The

strength of the relationship is simply the average of the perceived dominant and tonic strengths.

SCORING GROUPS

On the Difficulty of Grouping

Grouping in Musicat is different than in Copycat and other related projects because in listening there is time-based pressure to quickly form mental representations and one doesn't have the opportunity to carry out much retroactive modification of these quickly-formed representations. Fortunately, listeners make many quite sophisticated groups using relatively superficial cues. The key consists in having good heuristics and choosing the correct cues so that initial groupings are very good.

In music, an extreme change in one of many different parameters (such as pitch, dynamics, timbre, or duration) can be a strong indicator for forming a group boundary (according to GTTM's Grouping Preference Rule 3). In Musicat's microdomain, the only relevant factors in this list are pitch and note duration. Also, rhythmic gaps (relatively large time intervals between successive note attacks) are very strong clues to grouping boundaries. For this reason, Musicat's codelets suggest group boundaries at large pitch leaps, rhythmic gaps, and places where note density changes rapidly.

Grouping is a difficult problem in Musicat because there are in general many different possible groupings, and moreover, grouping can be hierarchical. For meta-groups, additional rules come into play. First, according to Grouping Preference Rule 4 in GTTM, higher-order group boundaries are suggested whenever the boundaries detected as described in the previous paragraph are relatively strong. A very large rhythmic gap suggests a group boundary at a higher level. Additionally, meta-grouping should occur in such a way that

components have roughly equal size, and there should only be a few of these components (2, 3, or 4, ideally, but not 1 (GTTM, Grouping Preference Rule 1), and probably not 6 or 7). Finally, meta-groups should be grouping parallel structures — this is where analogy comes into play.

On the Difficulty of Group Scoring

Of particular difficulty to implementing Musicat is that to compare any two rival groups we need a concrete, computational way to judge the relative strength of two groups. I have experimented with many possible approaches. The strength of a group is needed in several different circumstances in Musicat:

1. in fighting it out with another group to see which one will exist;
2. in calculation of the happiness of a structure (for structures, such as analogies, of which the group is a component);
3. in the probabilistic selection of whether or not to create a group;
4. in the probabilistic selection of whether or not to destroy a group.

Item 1 is the easiest to handle: in the case where we are comparing two objects, it is easy to let them “fight” by using a simple formula to add up points for various features that influence group strength, then to compare the two strengths, and flip a coin biased in a way that depends on the strengths of the two objects. In this scenario, the units or scale of the strength score is irrelevant; if necessary, we can normalize the score to a desired range by dividing by the sum of the strengths of the two groups. In contrast, items 2–4 involve an isolated group or a group in a situation where we would need its strength to be computed on some sort of absolute scale (between 0 and 100, say). Here are three possible methods to derive a meaningful group strength within the desired range 0–100:

1. Design the scoring function carefully as a weighted sum of range-restricted subcomponents, so that the minimum is 0 and the maximum is 100. This is the approach taken in Copycat.
2. Design a very simple strength function (such as the weighted sum of an arbitrary number of factors) that can become arbitrarily small or large. Track the statistics of recently scored groups, using a moving window of the past N strengths that have been computed. Normalize each calculated strength by computing its z-score relative to the statistics in the recent window.
3. Method 2 has the disadvantage of comparing apples to oranges; scores of a short group of just 4 eighth notes could be compared with scores assigned to a huge group spanning 16 measures. So method 3 is a refinement of that idea, where instead of statistically comparing all groups with each other in the same pool, many different moving windows of statistics are maintained — one for each particular context in which strengths are computed. The tricky part is deciding what the context is.

To implement method 3, we require explicit knowledge of which alternate structures the current proposed group is competing with. In Copycat, competing structures are destroyed because only one consistent view of the workspace is maintained at a time. In Tabletop, however, the notion of having alternative complete structures present in the workspace, with only one current Worldview, allows the program to store the history and the necessary statistics.

Although I used method 3 for some time I have returned to the simple approach of method 1, with a slight modification. I have found the current approach to be sufficient when I have a simple enough strength function.

Basic Formula

Each group is associated with a list of “group reasons” that describe why it is a group. This list is crucial to the group’s existence; without the list of reasons, the group would simply have a score of 0 and would vanish from the workspace. The problem with the simple weighted-sum formula (as used in Copycat) is that Musicat does not compute an exhaustive list of “features” of each group. Most typical AI approaches involving scoring possible representations, such as (Lartillot, 2004), require all features to be computed. In contrast, Musicat uses only those features that have been brought into perceptual focus by codelets, so each group might have a different collection of known features. This is especially true during the initial creation of a group, before additional codelets have run to examine it in more detail.

Musicat uses a slightly modified version of the simple weighted sum discussed above (method #1) to address the problem of scoring a group without computation of the exhaustive feature list. After computing the weighted sum of scores of all currently existing group reasons, we subtract a weighted sum of scores of group penalties, and then we run the sum through a squashing (*e.g.*, sigmoid) function to ensure that the result will remain in the range 0–100.

$$GroupScore = \sigma \left(\sum_{R_i \in GroupReasons} w_i R_i - \sum_{P_j \in GroupPenalties} w_j P_j \right)$$

with sigmoid function $\sigma(x) = 100 \left(\frac{2}{1+e^{-2x/100}} - 1 \right)$, and coefficients w_i and w_j determined simply by looking up the weight associated with each type of group reason or group penalty.

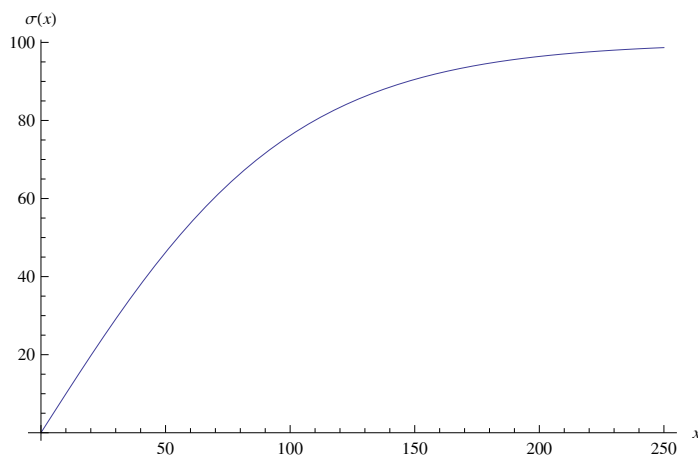


Figure 8.13: Sigmoid function.

The sigmoid allows us to be less strict about the weights in the linear sum: we can remove the constraint that the weights sum to 1. The downside is that if weights are too large, group scores might all become too high and might all squash down to essentially the same value near 100. Note that the function used here is just the right-hand half of the more typical S-shaped sigmoid. Often just one or two group reasons are enough to justify a group's existence, so I used a function that behaves nearly linearly near 0; there was no need to include the part of the S-shape that acts as a minimum activation threshold (this is used, for example, in many neural net models as well as in other FARG programs, such as Phaeaco).

SCORING SEQUENCES

Because a sequence is a subclass of the Group class in Musicat, scoring a sequence is the same as scoring a group. A sequences can have a list of group reasons just as can any other group, but it can also have a special sequence group reason with a score determined by the number of times the sequence's motif repeats and by the length of this motif.

SCORING ANALOGIES

An analogy in Musicat is a collection of relationships between two groups and their components. Analogies, like groups, have an associated score (or equivalently, strength). These scores are important for several different reasons, including:

- focusing codelet attention;
- determining happiness in the workspace;
- providing support to the groups involved;
- providing support to higher-level analogies;
- determining whether an analogy persists or gets destroyed in the workspace.

These ways of using analogy scores were important influences on my choice of how to score an analogy. I finally settled on the idea of a weighted sum of the following four factors (weights given in parentheses):

1. Size of the analogy (20%)
2. Completeness of the mapping (35%)
3. Strength of component relationships (35%)
4. How long the analogy has survived in the Workspace (10%)

Size of Analogy

Taking into account analogy size (in terms of number of measures) in the scoring of analogies is surprisingly subtle. On the one hand, small analogies may involve identical or nearly-identical components that are near to each other in time. Consider a sequence of three quarter notes, C-D-E, followed by three other quarter notes, E-F-G. It is easy for people to make an analogy between these two short groups. But now consider the larger-scale structure of the song “On the Street Where you Live”. Its first 16 measures are repeated nearly exactly,

except that the final two measures are modified to reach a stronger cadence. Is the analogy between the first 16 measures and the next 16 measures stronger than the analogy between C-D-E and E-F-G? The smaller analogy is more immediate and can thus be perceived more directly, whereas the longer analogy involves many more notes being mapped onto each other. Which should be scored more highly? We can also consider a non-musical analogy: in search of one, I looked out the window and noticed a tall tree covered with green leaves. I looked at one particular leaf and then saw a nearly identical leaf just below it, and of course, all the thousands of leaves on the tree were each very similar to each other, comprising a huge number of small leaf–leaf analogies. Then, “zooming out”, I noticed another large tree next to the first. It looked like the same kind of tree, and it was trivial to make an analogy between these two trees. In this case, I couldn’t even see most of the second tree; it wasn’t visible through the window, but I could see a few branches and many leaves. These two trees have quite different structures when I look at the detailed patterns of their branches and leaves, but they are clearly two highly analogous trees. Is the tree–tree analogy stronger than any individual leaf–leaf analogy?

The answer for Musicat is “yes”. Larger analogies are scored more highly than small analogies. Larger analogies are more difficult to form because they may involve more components, and because by definition they span larger segments of time. Musicat should pay attention to these large-scale analogies, should make sure they are given adequate codelet resources to flesh them out fully, and should recognize that the groups involved in a large-scale analogy are very strong.

Completeness of the Mapping

An analogy is a mapping between two groups and their components. Musicat enforces a time-ordering on the mapping so that the first element of group A must map onto

the first element of group B, the second element of group A onto the second element of group B, and so forth. Subject to this constraint, pairs of elements (one from each group) are put into correspondence. If any such pair is found for which no relationships exist in the Workspace to justify the correspondence, the analogy's score is reduced. The maximum possible score is reduced by the percentage of unmapped components found in the analogy.

Strength of Component Relationships

Just as a group's score is computed as a function of a weighted sum of group reasons, an analogy's score is derived from the relationships that form its basis. For each mapped pair of items in the analogy, a weighted sum of the relationships associated with the pair is computed and then this is run through a sigmoid function, just as in group scoring. Then the average such score for all pairs of analogy components is computed.

How Long the Analogy has Survived

Groups become more stable over time; as new notes are "heard", the age bonus associated with a group is increased. Since analogies are the most complex objects in the Workspace, they also need a mechanism to help them stabilize over time. Each analogy receives a bonus based on how much time (measured in codelets) has passed since it was created.

SCORING EXPECTATIONS

Groups

The strength of an expected group is based on the strength of the source group it was derived from. The base strength is multiplied by a weakening factor (such as 0.5) according

to the intuition that group expectations are more tentative than already-discovered groups. A bonus term based on the strength of any expected measure links associated with the group is added to the base strength score.

Analogies

As was the case with groups, the strength of an expected analogy is based on the strength of the source analogy, multiplied by a weakening factor such as 0.5. An expected analogy is further weakened if the size of the source groups is a “strange” size, in that the group size does not fit in well with that implied by the thicknesses of relevant barlines.

Measure Links

Expected measure links are assigned the same strength as the original measure links they are derived from.

Creating Structures with Codelets

This section describes in broad strokes how Musicat’s codelets work to create structures in the Workspace. For readers desiring more specifics without reading the source code, Appendix B gives more detail for each individual codelet type.

Each section below describes, for each Workspace structure, some of the codelets that directly contribute to creating the structure.

RHYTHM-BASED MEASURE LINKS

Measure links based on rhythmic similarity are among the simplest structures for Musicat to create. A **Measure Linker** codelet will examine two measures, selected probabilistically, with a preference for recent measures (unless otherwise specified by the

Measure Linker's parent codelet). The rhythmic similarity between the measures is computed and a link may be created, with probability of creation reflecting the similarity.

Measure Link Breaker codelets, on the other hand, examine measure links and may destroy the links, with a probability again based on the similarity score. However, these codelets leave certain links alone: if a measure is involved in only one link, that link will be spared destruction. Additionally, if a link is stronger than other links involved with either of these measures, the link is spared. Only the "weakest links" have a chance of being destroyed. Note that any measure can link to many other measures, and allowing a great many links to persist in the Workspace is generally not a serious problem because they do not interfere with other structures. This is in contrast with structures such as groups, which cannot overlap. An overabundance of measure links may reduce Musicat's focus, so Measure Link Breaker codelets are necessary. Removing measure links *quickly* is not necessary, however, so the codelets are carefully designed to only remove the weakest, most unnecessary links.

RELATIONSHIPS

Several types of Relationship objects can be created in the Workspace. Generally, each relationship type is created by a separate codelet designed to look for relationships of that type. Relationships generally involve either rhythm or pitch separately (although with analogy-based relationships, both may be involved)

Rhythm

The general "Look for Relationship" codelet tests two components from the Workspace, which can be measures or groups, and checks for an already-detected similarity between them. For example, it might find a rhythmic measure link, and then might create, with probability based on the link strength, a "Similar" relationship (or an "Identical"

relationship, if the rhythms are identical). More interestingly, it might find that an analogy between the two components already exists in the Workspace, and then create a “Similar” relationship based on the analogy. If these operations seem trivial, recall that relationship objects are important because they represent connections between two objects in a simple way that can be used to build up more complex analogies and other relationships. Generating a relationship from an analogy is precisely how Musicat implements “chunking” in memory.

The “Look for Starting Relationship” codelet looks for rhythmic similarity between the first parts of the components. It restricts the comparison to the first half of the shorter of the two components involved (with a two-measure maximum size) and computes rhythmic similarity. A “Start Similar” or “Start Identical” relationship may be created, again with probability based on the degree of similarity, and added to the Workspace.

Pitch

Now let us consider codelets operating in the domain of pitch, instead of rhythm. the “Look for Contour Relationship” codelet computes the melody contour of two groups and then may choose to create a “Contour” relationship linking the groups. As a reminder to the reader, contour relationships are displayed in the user interface as pink curves connecting the groups in question.

A rather different type of relationship is created by the “Look for Antecedent Consequent Relationship” codelets. This looks for two groups where the first ends on a pitch that might have a dominant chord function, and the second ends on the tonic. Of course, this is an extremely surface-level and simplified pitch relationship to find in a melody, and a true relationship between antecedent and consequent phrases has many more components. This is just one heuristic that Musicat uses in making connections between groups.

BAR LINES

The thickness of the bar lines in the Workspace is modified continually by **Measure Hierarchy** codelets. Each of these codelets selects a bar line at random from a window representing short-term memory, and chooses one of five actions at random to perform, using this bar line. Essentially, this means that each **Measure Hierarchy** codelet chooses the role of one of five completely different codelet types at run time; the five separate types are contained in the same **Measure Hierarchy** wrapper for historical reasons.

Bar-line thickness contributes to the strength of group boundaries. Conversely, strong group boundaries contribute to bar-line thickness. Each time Musicat encounters a new bar line as time moves forward, the line is initialized with the thickness expected if the music were following a typical binary pattern of alternating strong and weak measures, with this alternation happening recursively at higher and higher levels (longer and longer time spans, with each power-of-two length corresponding to a thicker bar line). Codelets can then modify this default thickness based on the endpoints of large groups in the Workspace.

Bar-line thicknesses are integer-valued, with the thickness of a bar line roughly corresponding to the expected length of groups having a boundary at that bar line. For a typical binary structure, the thickness would be the base-2 logarithm of the group length. For example, the thickness of the bar line between the first and second measures would typically be “0”, corresponding to minimum thickness, the thickness between measures 2 and 3 would be “1”, and the thickness between measures 4 and 5 would be “2”. The value 2 is the base-2 logarithm of the expected group length, 4 measures; a 4-measure group at the start of the piece would end just before this thickness-2 bar line.

Measure Hierarchy codelets must assign integer thicknesses to bar lines. However, it seems natural to program codelets to make gradual changes to thickness. Thus, the Workspace maintains a real-valued bar line thickness behind the scenes, simultaneously with

the integer thickness. Some codelets will modify the real-valued thickness, while others will make the more significant change of modifying the integer-valued thickness by adding or subtracting 1, thus moving the integer thickness closer to the real-valued, hidden version. This allows for gradual modification by many codelets, with sporadic jumps from one integer thickness value to another.

The five **Measure Hierarchy** codelet possibilities are:

1. move integer thickness closer to the real-valued thickness;
2. decrease real-valued thickness towards zero;
3. apply a boost to real-valued thickness, based on the size of a group ending before the bar line;
4. add random noise to the real-valued thickness;
5. if the bar line is relatively thick (thickness greater than 1), move the real-valued thickness of the next bar line towards 0.

GROUPS AND META-GROUPS

In Musicat, groups can rapidly come into and go out of existence. Recall that the strength of a group depends on the list of group reasons that support the group; at the moment of its creation, a group may just have one such reason, and hence a relatively low strength. Such a group may easily be destroyed by a **Group Breaker** codelet. A group may rapidly increase in strength, however, if codelets discover good reasons that are added to the group's description. Codelets that discover group reasons are described below. First, however, I answer the question of how groups initially come into existence.

Groups always consist of adjacent measures, so most codelets that create groups do so by examining adjacent measures and seeing whether a group is justified. The same principle applies to meta-groups: a meta-group must be composed of adjacent groups. The basic

grouping codelet is the **Proximity Grouper**. A **Proximity Grouper** codelet chooses two adjacent measures or groups in the Workspace and checks whether there are sufficient grounds to form a group (or meta-group) from the two elements. If a group is proposed, it is assigned a relatively weak initial strength of 50 (out of 100) and, like all new structures, must survive a standard probability test (a random number is generated and must be less than the structure's strength — in this case it amounts to a coin toss because the probability is 50%); moreover, the proposed new structure must compete with (and destroy) any existing incompatible structures before being added to the Workspace. If a new group is formed, an initial group reason is added to get it started with a non-0 strength. This is the very simple “Number of Elements” reason, which awards points to a group for having 2, 3, or 4 components, with a higher score given for 2 or 4 rather than 3 components. In this case, points are awarded because two adjacent elements have been grouped together,

The majority of elements available to the **Proximity Grouper** codelet are measures. Although it can create meta-groups, measures typically account for at least half of the possible group components, as we see if we consider the grouping structure of a melody as a complete binary tree structure — half the nodes in such a tree are “leaf” nodes, or measures in the case of a melody. Because of this distribution, **Proximity Groupers** don't try to create meta-groups as often as one might expect. To correct the problem, I designed a separate codelet type whose exclusive goal was to create meta-groups. The **Meta Grouper** codelet works much like the **Proximity Grouper**, but using groups as components. An optional parameter for the **Meta Grouper** codelet exists to specify a link to an existing analogy between adjacent groups. Other codelets may use this parameter to add pressure to form a meta-group based on an analogy. If an analogy has been specified and a meta-group is created, it will start with an “Analogy Support” group reason in its description.

Other codelets try to create groups of adjacent elements based on more than mere proximity. Each **Measure Sameness Grouper** codelet chooses a rhythmic measure link from the Workspace, with probability based on both the strength of the link and the recency of the most recent measure involved in the link. If the endpoints of the link are adjacent measures, the codelet attempts to group the measures together. If successful, the new group will start with either a “Components Identical” or “Components Similar” group reason. A group created in this way will thus start with a higher initial strength value than one created by a **Proximity Grouper**.

Another type of codelet, **Find Sequence**, looks for three or more elements in a row that have the same pitch contour and attempts to make a special Sequence group out of those elements. The final element in a sequence is allowed to be longer than the initial elements without penalty, reflecting the special case that in a sequence of three or more items, often the final item is extended to round out the musical phrase so that it has an even number of measures (recall Figure 8.8). Just as with groups formed by **Measure Sameness Grouper** codelets, sequences are created with an initial “Sequence” Group Reason, resulting in an initial group strength that is higher than for basic proximity-based groups.

ANALOGIES

As we have seen, groups in Musicat often are created by codelets for very mundane reasons (such as mere the proximity of two elements) and then are either destroyed or strengthened by other codelets. Analogies, on the other hand, require stronger support before they come into existence, and are given more time to solidify before they risk destruction by other codelets. In particular, the **Destroy Old Analogy** codelet attempts to destroy only analogies with low strength that were created several measures earlier.

The **Create Analogy** codelet is, unsurprisingly, the standard codelet responsible for analogy creation. Of course, to say one codelet is responsible is slightly misleading, as it depends on the prior actions of many other codelets to create groups and relationships of various sorts. The job of the **Create Analogy** codelet is to suggest a candidate analogy once the necessary components for a mapping between two structures are in place. The codelet may have been posted to the Coderack along with parameters specifying which two elements to include in a potential new analogy. If these were not specified, the codelet starts by choosing (probabilistically, as usual) an existing relationship that links an earlier group or measure to a recent group or measure in the Workspace. The earlier element is denoted the Left Hand Side (LHS) and the recent element is called the Right Hand Side (RHS).

Once the LHS and RHS of the potential analogy are identified, a temporary analogy object is formed. The subcomponents of each side are identified and any existing relationships between an LHS subcomponent and an RHS subcomponent are added to the temporary analogy. Not all relationships are valid: within an analogy, relationships compete with each other, much as a new potential group competes with any existing groups with which it would overlap. For example, if a relationship has been added to the analogy that maps element A of the LHS has onto element C of the RHS because both elements have the same pitch contour, it is not valid to add to the same analogy a second relationship in which element A is mapped onto element D of the RHS because they have a similar rhythm. The mapping must be consistent.

Once the relationship-based mapping from LHS to RHS is established, the **Create Analogy** codelet tries to add the analogy to the Workspace. The probability of adding the analogy is determined by the strength of the analogy. As is the case with groups, once the analogy is added to the Workspace, other codelets can attempt to strengthen the analogy.

During the process of creating a temporary analogy described above, **Create Analogy** codelets post many other codelets to look for additional relationships, groups, or analogies. For example, a lack of possible relationships to add to the mapping suggests places to look for new relationships; new codelets are posted to search in the right places. The **Add Relationships to Analogy** and the specialized **Add Nonrhythm Relationships to Analogy** codelets provide this focused search for relationships missing in the mapping structure of an analogy. Similarly, new **Meta Grouper** codelets are created to look for a higher-level group encompassing both the LHS and RHS.

Sometimes the placement of an analogy in the Workspace suggests another analogy. Specifically, when the LHS and RHS are separated in time, and a thick bar line occurs before the RHS, it is natural to expect a second, parallel analogy. The **Suggest Parallel Analogy** codelet looks for situations of this sort. Rather than creating an analogy itself, this codelet simply posts new **Create Analogy** codelets with parameters indicating where an analogy is hypothesized, as well as new **Proximity Grouper** and **Meta Grouper** codelets to create groups that would be necessary parts of the hypothesized analogy.

Analogies are the most complex objects in the Workspace. While watching Musicat run, we observed that sometimes a very promising (to human eyes) analogy forms but is later destroyed, despite its apparent strength. A certain amount of creation and destruction is fundamental to the FARG architecture. For a complex object, however, the difficulty of creation makes complete destruction and re-creation infeasible, especially given Musicat's real-time nature. Indeed, the program focuses on objects in recent musical time, so if a complex older object is destroyed, it is unlikely to be recreated before the musical elements in question have disappeared too far in the past to access.

The Tabletop program also faced the issue of the difficulty of re-creating complex objects. The solution adopted by French (1992) was to introduce the idea of a "Worldview",

distinct from the Workspace (see Chapter 2). I used a similar strategy in Musicat. The **Store Strong Analogy** codelet selects a recent strong analogy and has a chance to add the analogy (including all its consistent groups and relationships) to a permanent record for later recall in case it is destroyed. The **Resuscitate Analogy** codelet can attempt to bring back a previously-discovered analogy from memory all at once. This allows a complex analogy to be revived when the normal mechanisms of gradually building-up groups and relationships might not be sufficient given the real-time pressure of the listening domain.

EXPECTATIONS

Codelets related to expectations can be divided into two types: those that create expectations, and those that use the expectations later on to influence grouping and analogy-making in the Workspace.

The **Generate Expected Group Copy** codelet creates a basic kind of expectation: its job is to add an expected group to the Workspace. It does this by finding a strong group that has ended just before a relatively thick bar line, and attempts to generate an expectation for a group of the same size to come after the bar line. The structure of the group is also expected: any subgroups and measure links associated with the group are also turned into future expectations. In addition, a high-level analogy is suggested (*i.e.*, added to the list of expectations) to link the presently-existing source group with the future expected group. The **Generate Expected Analogy Copy** codelet does the related job of generating an analogy that is expected to occur in the future, between two expected groups. It does this by looking for expected groups that have already been generated by the **Generate Expected Group Copy** codelet, and then building expectation for analogies between the groups if there were analogies between the relevant source groups that generated the expectation. Basically, these

two codelet types work together to create expected future groups and expected analogies between future groups.

Several more codelet types use the generated expectations to try to create real structures when the time comes. The **Suggest Link From Expectation** codelet does this for measure links. This codelet always focuses on the most recently heard measure, and looks for expected links that end on this measure. If such links are found, one is selected at random and the codelet attempts to build a real link between the suggested measures. The **Suggest Group From Expectation** codelet selects an expected group, ignoring any groups which exist entirely in the future. If at least the first measure of an expected group is a measure that has already been “heard” by the program, then the expected group can be used to create a real group. Note that when such a group is formed, it often contains just a single measure. For example, if an expected group spans measures 9–12, once measure 9 is heard, a codelet might try to form a real group starting on measure 9, and only containing that single measure. Another type of codelet, **Extend Group Right**, is responsible for attempting to extend groups to the right, especially when the extension is suggested by the existence of an expected group extending to the right. Finally, the **Suggest Analogy From Expectation** selects, at random, an expected analogy in which the measures that would make up the right hand side of the analogy have been heard (*i.e.*, the entire analogy must exist in the “heard” portion of the Workspace, not in the future.) This codelet simply posts a **Create Analogy** codelet to the Coderack to attempt to build the specified analogy.

CHAPTER NINE

The Evolution of Musicat

In the introduction to this dissertation I wrote that Musicat is a model of music listening. But it is misleading to call it simply *a* model, in the singular, because it is actually a work in progress and has evolved significantly from its original form. The original versions of Musicat shared the same underlying FARG architecture, but share surprisingly little code with the current program. Most significantly, I rewrote the bulk of the program starting in September 2011, and it reached its current state in the early summer of 2012. The listening performances of Chapters 5–7 and the description of the program’s architecture in Chapter 8 are based on that latest version. This chapter, however, covers the previous versions of Musicat. Many of the ideas from previous versions did not make it into the latest version, but I think some of them are important and should be resurrected in future incarnations of Musicat.

In addition to showing old ideas in prior versions of Musicat, this chapter also highlights some of the problems that plagued Musicat in the past. To be sure, it has plenty of problems in its latest version, but early versions had even more issues, and a discussion of those problems may prove even more illuminating than any of Musicat’s successes.

A Brief History of Musicat

Because Musicat has been evolving gradually over time as I have tried out new ideas and focused my energy on different aspects of the program, it is difficult to clearly delineate different versions of the program. However, I can describe several time periods of development and mention the most salient ideas and issues of those times. In addition, there were a few major architectural changes that took place, and those turning points are key moments that invite the use of different names for the program. Some of the most significant time periods and architecture changes are listed below:

- Spring 2004–Spring 2008: Initial work. Hierarchical, Schenkerian, based on Larson’s multilevel Seek Well model. I call the program from this time “**MusicatH**”, for “Musicat with Hierarchy”.
- Fall 2008–Fall 2009: Removal of hierarchy (starting here, the program is named “**MusicatW**”, for “Musicat without Hierarchy”). Addition of more complicated rhythms. Development of a detailed user interface for adjusting parameters. Attempts at machine learning of parameters. Addition of “perspects” to notes and groups.
- Spring 2010: Semester in Paris. Increased use of motivic memory, and spreading activation to similar motifs. Improvements to user interface, including visualizations of codelet distribution on the Coderack. Some effort to handle larger-scale melodies such as “Mademoiselle de Paris” (previously, only 4-measure long melodies had been used extensively). Groups are forced to start and end at bar lines instead of mid-measure.

- Fall 2010–Spring 2011: More emphasis on the importance of chunks of music with powers-of-2 lengths. A new analogy data structure was added.
- Fall 2011: A new emphasis on the primacy of rhythmic structure in musical listening. Also, previous unresolved problems led me to try two radical new ideas:
 - **BinaryCat**: I wanted to focus on grouping and analogy-making at the measure-level and higher, so I toyed with the idea of representing each measure with a high-dimensional binary vector, inspired by *Sparse Distributed Memory* (Kanerva, 1988). This was just a fleeting experiment, but it led to:
 - **RhythmCat**: This was a complete rewrite of Musicat, intended to focus only on rhythm. Melodies were replaced by purely rhythmic patterns; notes no longer had pitches. Grouping and analogy-making were dependent exclusively on rhythm. Surprisingly, this experiment led to the current version of the Musicat.
- Spring 2012: RhythmCat evolved into the latest version of the program, simply called **Musicat**. I added pitch information back into RhythmCat, resulting in a program that “listens” to pitched melodies but generally treats rhythm as the primary musical feature. “Plug in” for Visual Studio was used to give me faster feedback about the program’s behavior as I was making architectural experiments.

Versions of Musicat

This section describes the previous versions of Musicat at significant stages of its development. Three major versions are described: “Musicat With Hierarchy” (“MusicatH”), “Musicat Without Hierarchy” (“MusicatW”), and “RhythmCat”. Afterwards, I make a few comments about how RhythmCat was converted into the latest version of Musicat. But first, a few comments on my early work on modeling the listening that happens at the very beginning of a melody.

TWO-NOTE MODEL (KEY AND METER INDUCTION)

My very earliest work on Musicat had a very different focus from all subsequent versions: the program was based on the FARG architecture, and it simulated melody perception but only for extremely short melodies. Specifically, my first goal was to make a program that was able to infer (or at least guess at) the key and meter of a melody very rapidly, after hearing just the first few notes. Steve Larson suggested this project to me in 2004. Reducing the size of the melody under consideration even further, he gave me the preliminary “assignment” of writing a program to implement what he called the “Two-note Model”. Given just the first two notes of a melody, the Two-note Model would form a tentative idea about the key and meter of the melody. Then, additional notes would be given one at a time and the program would change its initial idea of the key and meter if necessary. For example, given the “Triplet Melody” discussed in Chapter 3, the desired program would initially hear the melody in 4/4 time in C major, but after the fourth or fifth note, it would change its interpretation to 3/4 time (still in C major). This model was related to Larson’s earlier work in which he asked music students to sing continuations of two-note-long melodic fragments (Larson, 2004).

I started development on this model, which eventually led to the program “Musicat Without Hierarchy”, discussed below. In parallel, I also ran a pilot study with human subjects (see Appendix A) to learn more about how people infer key and meter at the very start of a melody. After spending a significant amount of time on such issues, I came to realize (thanks to discussions with Douglas Hofstadter) that the activity of inferring key and meter was not at all central to our vision for the Musicat program, even though I think the FARG architecture is very well suited to solving this modeling problem. Instead of having Musicat spend time focusing on just two or three notes at the start of the melody, we wanted it to make musical analogies of various sizes and to listen to entire melodies of 16, 32, or possibly many more measures. In 2008 I stopped working on the key- and meter- inference part of the program and turned to larger-scale issues. Throughout Musicat’s development, this trend continued: I made significant progress by focusing my attention (and the program’s attention) on ever larger structures, as well as by devoting ever more attention to the idea of musical analogy-making.

MUSICAT WITH HIERARCHY (MUSICATH)

The first version of Musicat, counterintuitively, was in many ways the most sophisticated. Although its origin was in the Two-note Model program described above, I designed it on the basis of several discussions with Steve Larson about how his Multilevel Seek Well program could be converted into a program based on the FARG architecture. Musicat with Hierarchy (henceforth “MusicatH”) was focused on generating melodic expectations for the next note (or the next several notes), using a Schenker-inspired multi-level analysis as well as small-scale note groupings.

Main Features of This Version

In order to make MusicatH more concrete and to demonstrate the main features of the program, I present a sample screenshot from June 2008 (I removed the explicit levels of hierarchy from the program shortly thereafter), using the melody in the following figure:



Figure 9.1: Opening of “Stückchen”, from Schumann’s *Kinderszenen*, Op. 15.

The melody of this Schumann piano piece was not a valid input for an early version of MusicatH, because only quarter notes were allowed, and this melody has a half note in measure 2. The program initially could “listen” to melodies from the Seek Well microdomain only (although I later added support for half notes, whole notes, and some dotted notes). Accordingly, I modified Schumann’s melody by breaking the half note into two quarter notes (rests were not allowed; otherwise, I might have used a quarter rest on beat 4 of measure 2).

The next figure illustrates the kind of output that I had hoped this version of the program would generate. The figure shows the program during a run on the modified melody, but note that this is partly a sketch: the screenshot was modified slightly to make a *mocked-up* image of the desired user interface. The *groups* in the image were created by MusicatH, but the group *links* and “active motives” were added by hand).

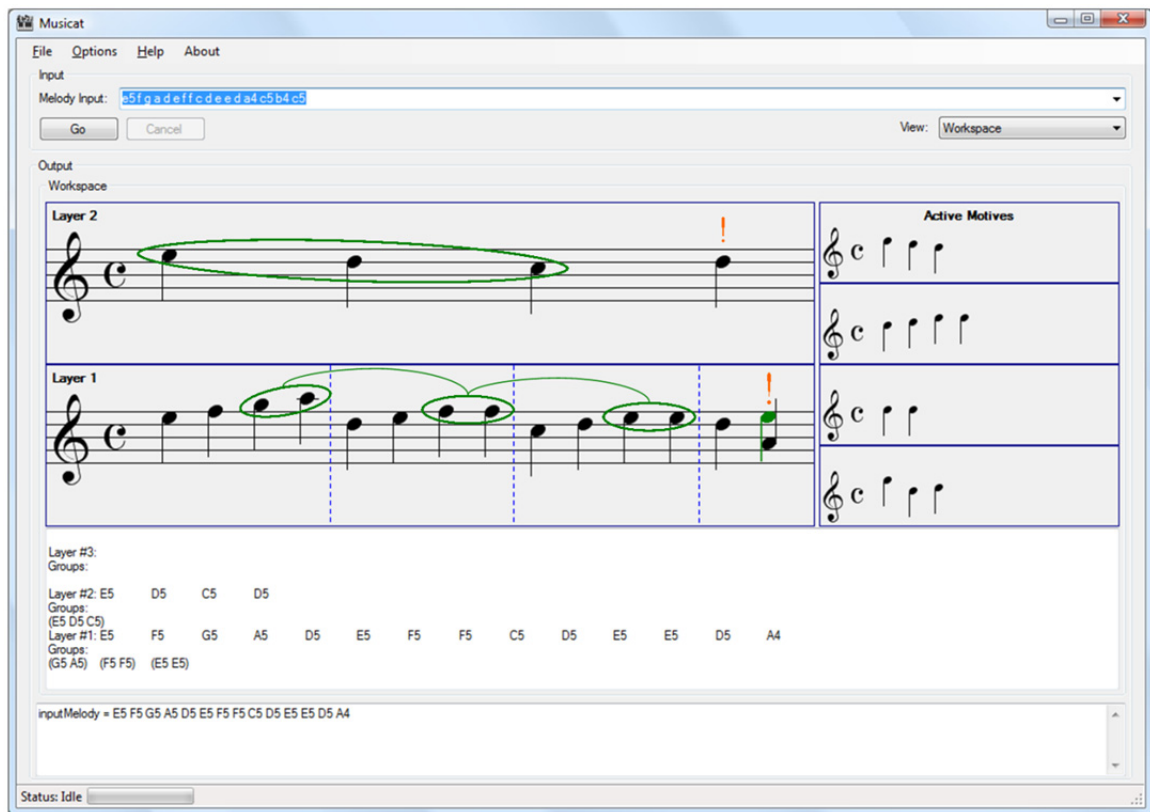


Figure 9.2: MusicatH listening to a modified version of “Stückchen”.

Layer 1 in the figure contains the melody that the program is listening to. Layer 2 is generated by the program as it runs, and it consists of notes that have been “promoted” from Layer 1 because of perceived structural importance. The program theoretically had the ability to generate additional layers at even higher levels, but there were enough problems even in the two-layer case that I usually restricted the number of layers to two.

The green ellipses represent groups, just as variously-colored ellipses do in the current version of Musicat. MusicatH can make groups in any layer, and they consist of collections of adjacent *notes*, rather than of adjacent *measures*. There is no restriction for groups to start and end at measure boundaries. There are no meta-groups in this version of Musicat, but groups can overlap temporally with groups in a different layer (*e.g.*, in the figure, the first two groups in Layer 1 overlap in time with the group in Layer 2). Groups in the same layer cannot in

general overlap, although yellow groups, indicating pitch repetition, are an exception: a green group can overlap with a yellow group.

MusicatH is especially focused on noticing linear patterns and exact repetitions of pitch. For example, the groups formed in the figure are all linear motions, such as the (E–D–C) group in Layer 2 and the (G–A) group in measure 1 of Layer 1, or pitch repetitions, such as the (F–F) and (E–E) groups in Layer 1, measures 2 and 3.

Note expectations are generated much as in Larson's Multi-Level Model: the program makes predictions at a higher level (Layer 2) and then these predictions are used to make smaller-time-scale predictions on a lower level (Layer 1). Predictions are visible in the figure as green-colored notes. When a note arrives that is not expected, the program draws an orange exclamation point to indicate surprise. For example, in Layer 1, the program expected the note E, because it had noticed the pattern established in previous measures of notes moving up by scale step after the initial note of each measure. However, the note A occurred instead, so it drew an exclamation point above the A. Similarly, one quarter note earlier, in Layer 2, the program expected the note B to occur because of the perceived descending scale group (E–D–C). (The green B is not shown in the figure because only the most recent wrong prediction, in this case the E, is displayed.) One might recall that Layer 2 is made of notes that have been perceived as structurally important, and one might thus wonder how a prediction can either come true or be denied in Layer 2, when the actual notes “heard” by the program are added to Layer 1, not to Layer 2. Well, in this case the note D was perceived in Layer 1, and subsequently it was promoted to Layer 2; this promoted note D was a surprise since the program expected B; therefore, the program indicated its surprise by the orange exclamation point above the D in Layer 2.

The division of the melody into separate measures (*i.e.*, the placement of bar lines) was *inferred* by the program, although this never worked very well. However, in the particular run from which this screenshot was taken, the dashed blue bar lines were *given* to the program along with the input; it was not attempting to generate bar lines during this run. Similarly, the key of the melody was supposed to be inferred, although development of this aspect never got very far; in later versions I provided the key as well as the bar lines.

Melodic motifs were not generated by this version of the program, but this mocked-up screenshot shows that they were intended to be displayed in a box at the right (labeled “Active Motives”). These are short melodic shapes that the program has heard as being salient. For example, the Layer 1 groups in measures 2 and 3 exhibit a 2-quarter-note pattern consisting of a simple note repetition. This pattern, if the program had actually noticed it, would have been designated as a motif: it is the third entry from the top in the “Active Motives” display.

MusicatH had a Slipnet modeled after the Slipnet in Copycat. This Slipnet included concepts such as “step”, “leap”, “ascending”, “descending”, “pitch repetition”, “linear motion”, “tonic”, “subdominant”, “dominant”, and a collection of pitches such as “C”, “D”, “F#”, and “Ab”. However, I didn’t figure out a cogent way to coordinate the standard Slipnet mechanisms with the constant flow of time in the music domain, so I eventually (but reluctantly) removed the Slipnet. See Figure 9.3 for a screenshot of MusicatH’s Slipnet.

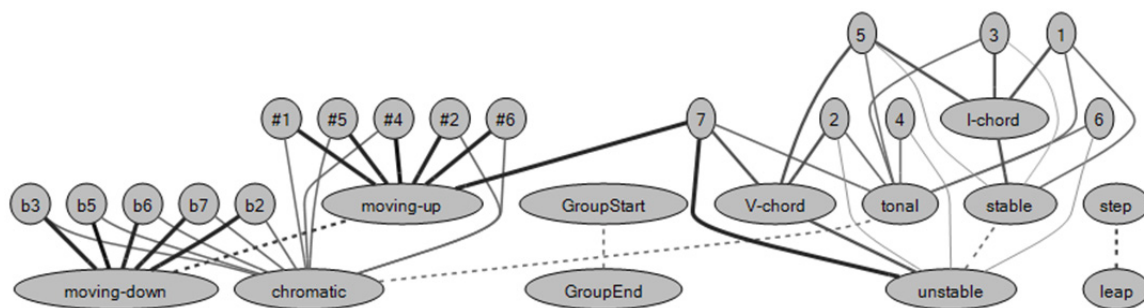


Figure 9.3: MusicatH's Slipnet.

An obvious difference between this version of the program and the most recent one is the use of standard music notation. Quite a large percentage of the code in MusicatH was devoted to the proper display of music notation. Correctly laying out elements of music notation is quite complex, even in the restricted case of Musicat's melodies. Writing my own display code allowed me to draw group ellipses that enclosed notes in an elegant way; see, for example, the way in which the large group in Layer 2 is tilted to slope downwards at the right end of the group, whereas the small group in Layer 1, measure 1, slopes upwards. Out of both speed and complexity considerations, I stopped drawing the notation so carefully in the more recent version of Musicat. See (Byrd, 1984) for more details on the fascinating topic of computer-generated music notation.

Example Runs with This Version

The snapshot above shows a fairly *good* run of the program (all things considered), and hence it masks a variety of problems. For instance, the program would often make groups in Layer 1 that seemed much less analogous to each other than these do. All three small green groups in the figure are similar to each other, as they start on beat 3, end on beat 4, and form the end of an ascending sequence, but this kind of similarity is in fact not very common.

A more obvious problem in this run is that many notes remain ungrouped. This is a serious problem — the first two notes, C and D, are not grouped, even though the promoted version of the C in Layer 2 is part of a group. In general, for most runs, many notes never become part of a group; this is because MusicatH did not feature a pressure pushing hard for each note to be seen as a member of some higher-level structure.

To give a better picture of how the program worked and to illustrate more of the problems it had on even the simplest of melodies, I present a collection of screenshots from MusicatH with minimal commentary. But why, one might ask, am I bothering to show and analyze results of a program that didn't work very well? The answer is that it is important to understand the extreme complexity of the domain. Musicat could not exist in its present form without our first having profoundly grappled with the many problems that came up in earlier implementations.

During the development of MusicatH, I spent most of the time experimenting with the Schumann melody from the previous example, with a “Triplet Scale” melody, and with a few others. In this section I will use just a few of these melodies as examples, as well as melodies such as “Twinkle, Twinkle, Little Star” and “Good People All”, which I hadn't run the program on during development.

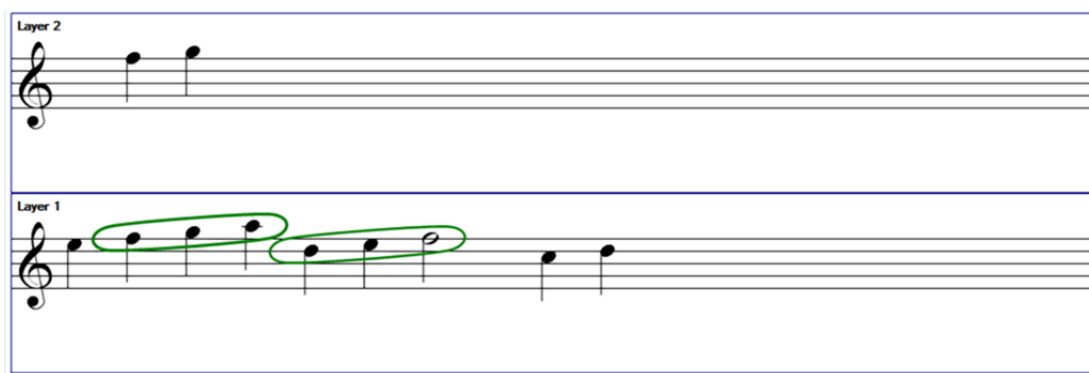
“Stückchen”, by Schumann



Figure 9.7: Triplet Scale.

When people initially hear a melody, they typically assume it is in a duple meter like 2/4 or 4/4, but given contrary evidence, they may quickly shift to an alternate metric interpretation. In this melody, people will switch to a 3/4 interpretation at some point (even if the notes are all played with exactly the same volume and articulation):



Figure 9.8: Triplet Scale, heard in 3/4 meter.

Let's take a look at how MusicatH "heard" this melody.

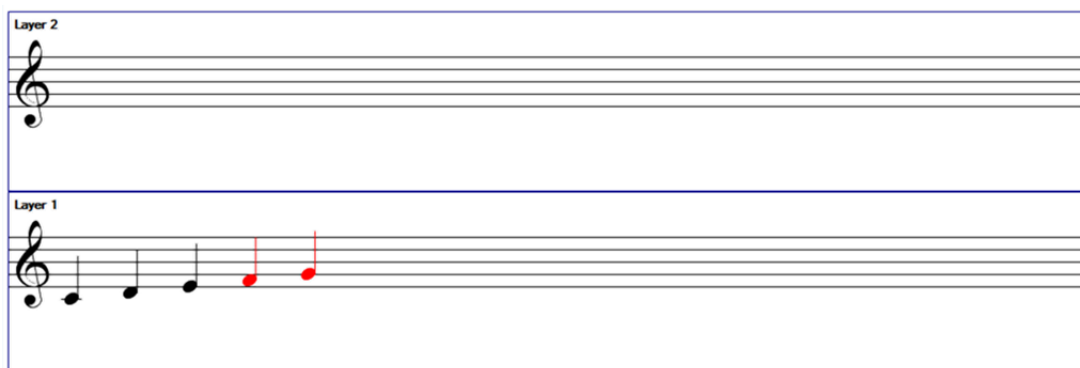


Figure 9.9: Triplet Scale, run 1.

After the first three notes (Figure 9.9), the program has generated an expectation (shown in red in this and the following diagrams) that makes sense: the scale is expected to continue up to G.

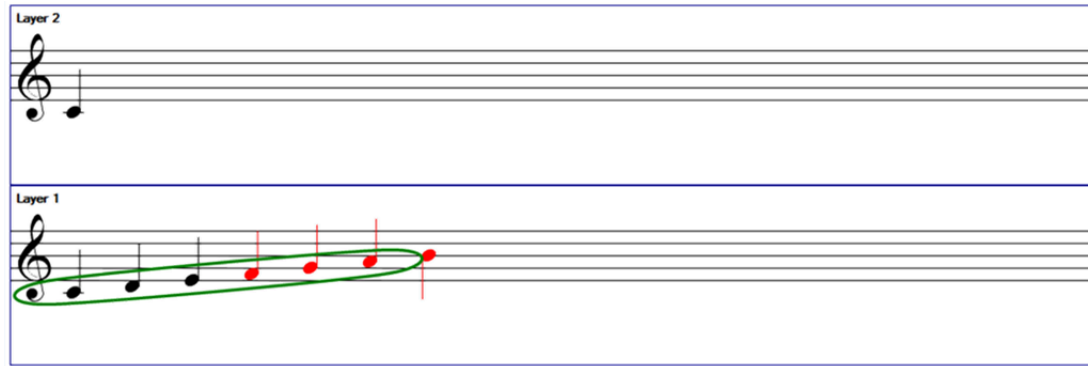


Figure 9.10: Triplet Scale, run 1.

A moment later (Figure 9.10), the prediction has been extended all the way through B, and the first six notes (not including the B, strangely) have been grouped together.

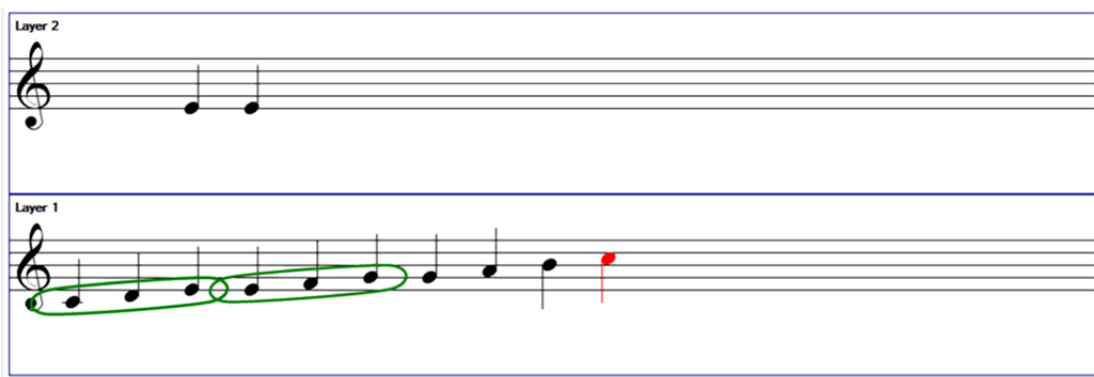


Figure 9.11: Triplet Scale, run 1 (end of run).

At the end of the run (Figure 9.11), the program's earlier predictions have been replaced with the actual notes. The first six notes have been grouped in way I expected: two groups of three notes each. The final three black notes, unfortunately, were not grouped. The program generated the obvious prediction for the scale to continue up to C (however, the final C wasn't included in the melody given to the program in this run.)

Layer 1, aside from the missing final group, showed a reasonable grouping structure. Layer 2, however, makes absolutely no sense. Ideally, the layer would contain an ascending

arpeggio, C, E, G, with a red expected note C at the end, just as in Layer 1. Obviously, the multi-layer mechanisms in MusicatH didn't work well.

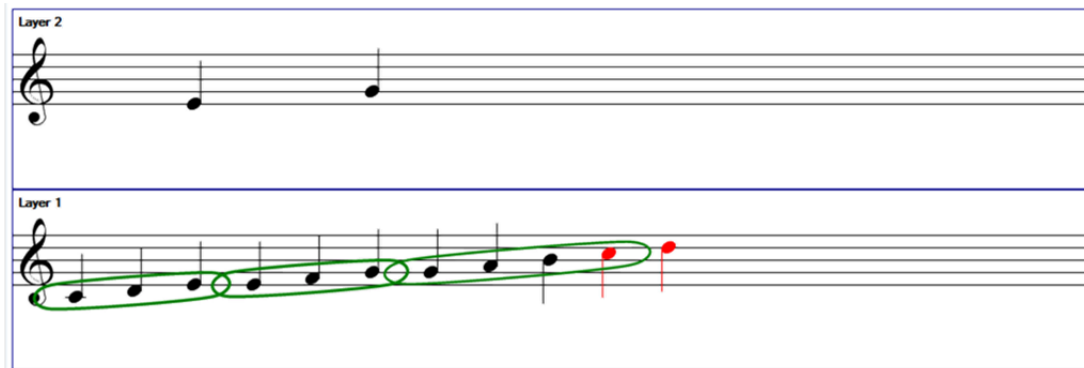


Figure 9.12: Triplet Scale, run 2.

In another run, the grouping structure in Layer 1 was quite reasonable. The final expected D, however, belies a lack of understanding of the melody (a repeated C would fit much better, if the program had understood the melody as based on an arpeggiation through a C-major triad in Layer 2). Layer 2 was better than in the previous run: it was on the right track, but the initial C was conspicuously missing, and thus no arpeggio was “heard” in the layer.

These two runs showed problems in Layer 2, but they did a passable job on Layer 1. However, many runs are quite worse. The next two show significant problems in both layers.



Figure 9.13: Triplet Scale, run 3.

In this third run, the grouping structure in Layer 1 seems almost entirely random. The program has made a pitch-repetition group (colored yellow) around the first two E's (such groups were missing in the previous runs — also notice that yellow and green groups are allowed to overlap), but the green group (E–E–F) makes little or no sense. Layer 2 also makes no sense; it is almost a copy of Layer 1 for the first six notes of the melody.



Figure 9.14: Triplet Scale, run 4.

In this final run, just one group has formed, in Layer 1. Layer 2 has the expected three notes C–E–G, but also an additional F that doesn't fit in. Altogether, very little structure has been found, and what little has been found makes very little sense.

Twinkle, Twinkle, Little Star

For this melody, the program notices many of the note repetitions — a very salient feature of this melody — but fails to understand anything much else of the structure.

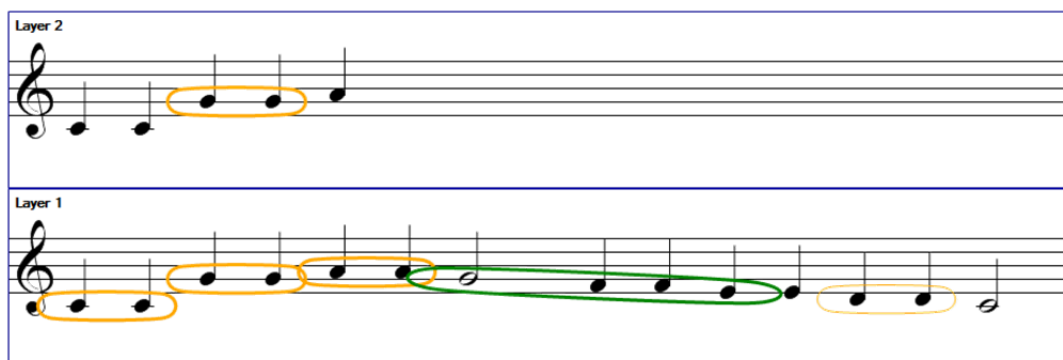


Figure 9.15: Twinkle, Twinkle, run 1.

The yellow pitch-repetition groups that exist are good, but there are some missing (such as around the two F's). Layer 1, again, doesn't make much sense. The green group captures some of the descending scale, but it's a very strange segment, going from the half-note G down through the first E. It is strange that it didn't continue to the final C, and also quite strange that the group *starts* on a long-duration note (the half note) that, to a human listener, *ends* the previous phrase. The program has noticed the linear descent, and it has essentially ignored the significance of the half-note duration of the note G. The pause in motion created by the half note strongly influences group-creation for a human listener and suggests that a group should end just after this pause. The problem of ignoring the group boundary implied by the half-note was a central issue for the next version of the program (Musicat without Hierarchy), when running on the melody "On the Street Where You Live".

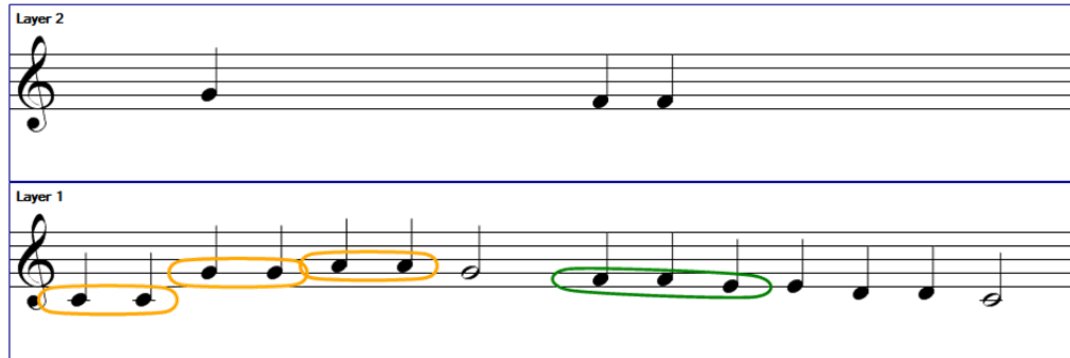


Figure 9.16: Twinkle, Twinkle, run 2.

In a second run (Figure 9.16), similar structures formed, but the green group was shorter...

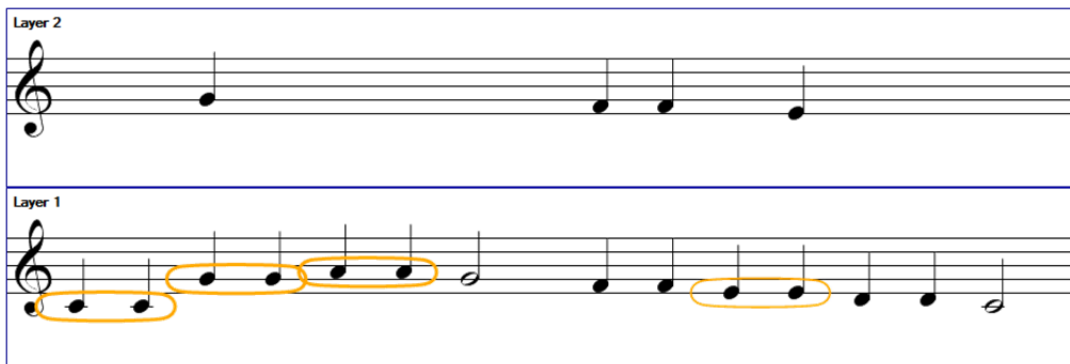


Figure 9.17: Twinkle, Twinkle, run 2, a moment later.

... and just a moment later (Figure 9.17), it had disappeared entirely (although another yellow group has formed). This gives a taste of another problem that plagued the next versions of Musicat (and that continues to cause trouble in the current version): the program is flexible in trying out different structures, but far too often it seems to “flail around”, trying out various small structures without seeing the larger structure. In this case, the program never notices the long descent from F (or even G) to the final C: it tried out the short (F–F–E) group and then gave up, thus failing to perceive any good large structure.

Good People All

A positive thing about the early versions of Musicat was that they could make groups at any point. In “Good People All”, this feature allows one group to form that is impossible in the current version of Musicat:

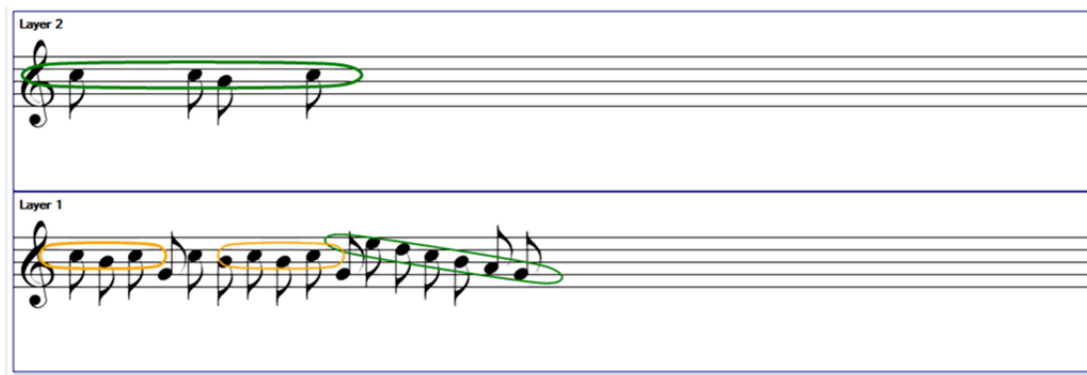


Figure 9.18: Good People All (first two measures).

The green group in Layer 1 shows that the final six notes of this excerpt have been heard as a descending scale, and this segment is separate from any other groups. Unfortunately, the program did not make any groups for the first ten notes, aside from the two yellow pitch-similarity groups, which exclude not only the non-C notes, but also the third C, which is ungrouped. The green group in Layer 2, however, does appear in the right place — I might hear the first ten notes as forming a single group, and the program has picked the three most important “C” notes (in my own hearing) to promote to Layer 2. It is unfortunate that a stray B was promoted as well, and also that the extremely salient E, which starts the descending scale in Layer 1, was not promoted to the top layer.

On the Street Where You Live

I will show more examples of this melody in the section on the next version of Musicat, but it is especially instructive to see a problem that crops up in this early version of the program on the first four measures (notice that none of these examples have been more than a few measures long, because the program did extremely poorly with longer ones):

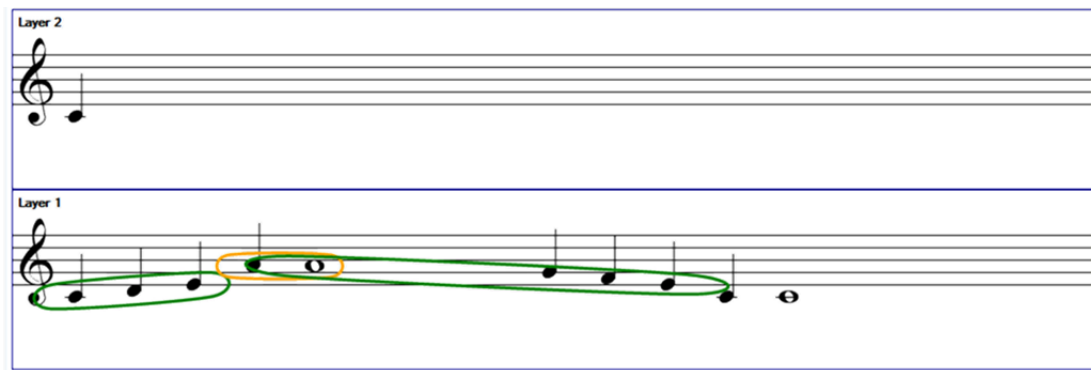


Figure 9.19: *On the Street Where You Live*, run 1.

The first run on “Twinkle, Twinkle” (Figure 9.15) exhibited a very similar problem, but it is more pronounced here: the second green group starts, quite unnaturally, on a long note, the whole note A (the program’s display routine draws the group so that it extends too far to the left and touches the quarter note A, but for the program, the whole note A is indeed the first note of the group). This problem of starting a group on a long note was quite persistent and was very hard to eradicate for quite some time during the development of Musicat. Why did the program create this bizarre, long group that so obviously spans the huge temporal gap between the whole note A and the following G?

There are two obvious competing factors in the grouping here. First, the whole note creates a pause in the previously established rhythm of note attacks (there had been an attack once every quarter beat, and suddenly there are four beats between attacks). This pause creates a pressure for a group boundary to form after the long note — this group boundary is

obvious to me, and I believe that any human listener would form the boundary at this location. A second relevant pressure, however, exists in the program: it favors making groups exhibiting stepwise motion, and longer groups of this sort are assigned higher strengths. The stepwise descent from A to E is long indeed, and hence results in high strength value. To human ears, however, the temporal gap in attack onsets caused by the whole note is much more salient, and the program's grouping seems ridiculous.

It would be possible to adjust the relative strengths of these two pressures to fix this problem, but there is another factor that helped even more to reduce this problem in subsequent versions of Musicat: analogy. The first five notes should be heard as *analogous* to the next five notes (*i.e.*, the first two measures are analogous to the second two, if we have bar lines and a 4/4 time signature). This parallelism makes the grouping boundary after the whole note even more obvious, because the three quarter notes G–F–E are so easily mapped onto the initial notes C–D–E.

The next example (Figure 9.20) is of a moderately good run that illustrates a subtler problem in the same melody:

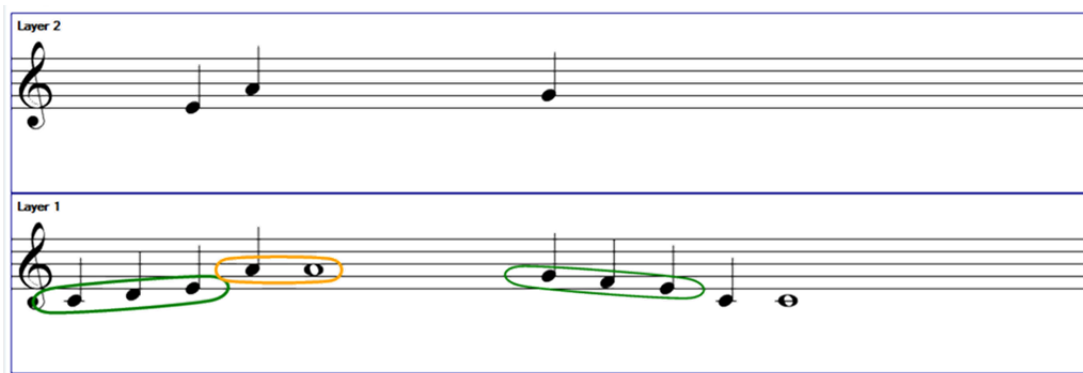


Figure 9.20: On the Street Where You Live, run 2.

Both of the three-note scale passages have formed groups (shown in green). The first note repetition has also been identified (shown in yellow). Unfortunately, the matching final note-repetition (the two C's) did not yield a group. Layer 2 has picked out some potentially-important notes, but absolutely no large-scale structure has been discovered.

Perhaps most importantly, in terms of the development of Musicat, notice that this version of the program doesn't make group links in the display. Behind the scenes, while the program is running, codelets do use the existence of one group (such as the first green group) to inspire a search for similar groups (such as the second green group). However, once a similar group has been formed, no analogy is perceived between them. The loss of this linking information severely crippled this version of the program. If an analogy had been established between the green groups, it could have guided the search for a *second* yellow group, and a run that started as shown above could have resulted in a complete grouping structure, at least in Layer 1.

I gave examples of the centrality of analogy-making at all levels in Chapter 4, but ironically, during Musicat's development, analogy wasn't a large part of the program until relatively late in its development. Explicit group links, which could indicate that two groups were similar in some way, were part of the program already since 2008 (Musicat Without Hierarchy), but a more structured "analogy" object, which could describe the mapping between two sides of a correspondence, was not added to the program until early 2011. In retrospect, this sort of analogy object — one that could collect together a set of relationships amongst its components — seems to have been one of the most crucial missing pieces in these early versions of Musicat.

MUSICAT WITHOUT HIERARCHY (MUSICATW)

It is obvious from MusicatH's results in the previous section that Layer 2 wasn't useful, mainly because there was not enough structure perceived in the notes that were promoted to it. While I believe that this multi-layer idea has much merit, it would have required significant changes to make it work well. Rather than spending more time on that aspect of the program, I decided to remove the multiple layers of hierarchy. The resulting version, Musicat Without Hierarchy (henceforth "MusicatW") may have lacked a Schenkerian-style hierarchy, but in its place I added the ability to form meta-groups, which provide a different way to look at hierarchical structure. Some aspects of musical structure, such as linear motion on a large time-scale, are harder to capture without a multi-layer analysis, so I think it would be useful to re-examine this idea in the future, once the program works much better on smaller structures.

MusicatW evolved in design quite a lot over the two-and-a-half years of its development, starting with the removal of hierarchy layers and ending with the addition of a new "analogy" object type. I will describe the essential elements of the final form it took in 2011 before the most recent major redesign.

Main Features of MusicatW

MusicatW was the most complicated version of Musicat, because over the course of its development many different features were added, both in the model's architecture and in the user interface. Figure 9.21 is a screenshot of MusicatW in action for nearly the same melody as was used to introduce the previous version (the only difference is that here, the F in measure 2 is a half note instead of two quarter notes, as in the later runs of MusicatH in the previous section).

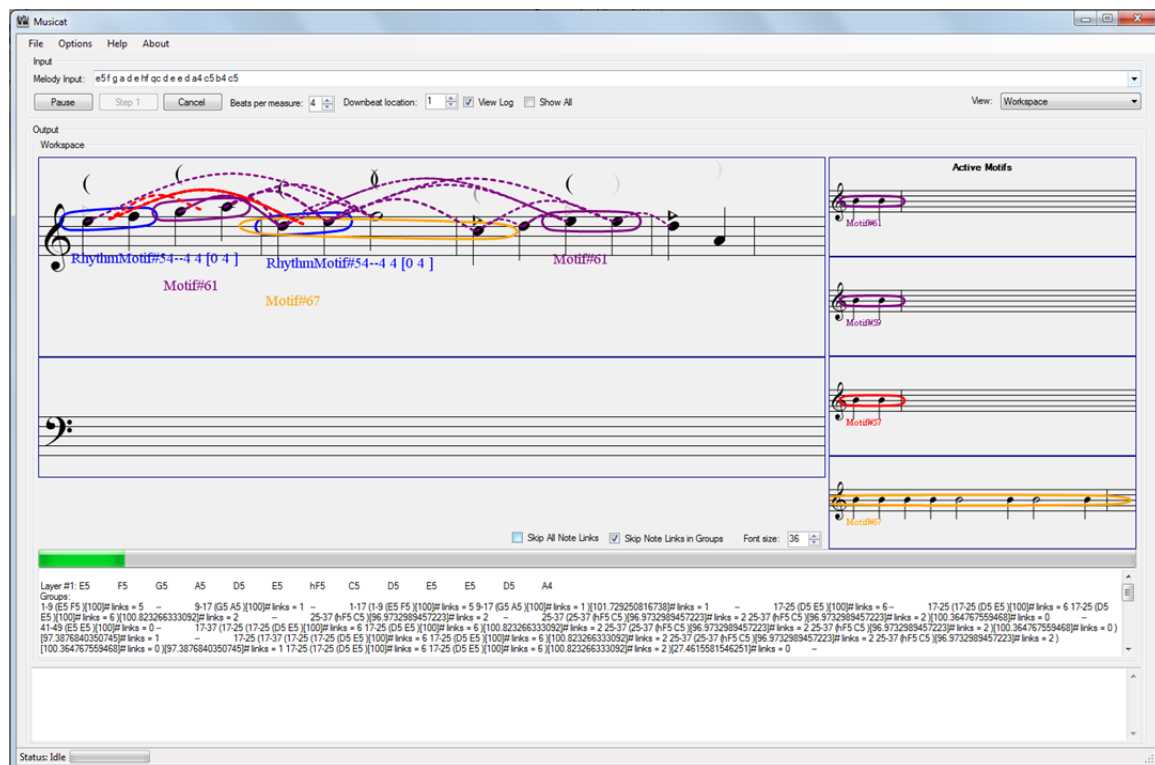


Figure 9.21: MusicatW running on “Stückchen”.

Notice a few brand-new elements here: there is only one layer — the melody itself, in the top staff, but there is also a staff displaying an optional bass line (not used in this example). There is a short green bar underneath the bass staff. The length of the green bar indicates the current global temperature of the Workspace.

In this screenshot, the “Active Motifs” are actually generated by the program, as are all the links between structures in the Workspace. Dashed *red* arcs connecting *groups* are group links, whereas dashed *purple* arcs connecting *notes* are note links. In MusicatW, note links take the place of the yellow pitch-repetition groups generated by MusicatH, and are also more general (notes heard as similar do not need to share the same pitch: a D in one measure might have a note link connecting it to a G in another measure. Note links tend to

clutter up the image, so the following figure shows the same run from a moment earlier in the run, without the note links. I have also zoomed in to display only the Workspace.

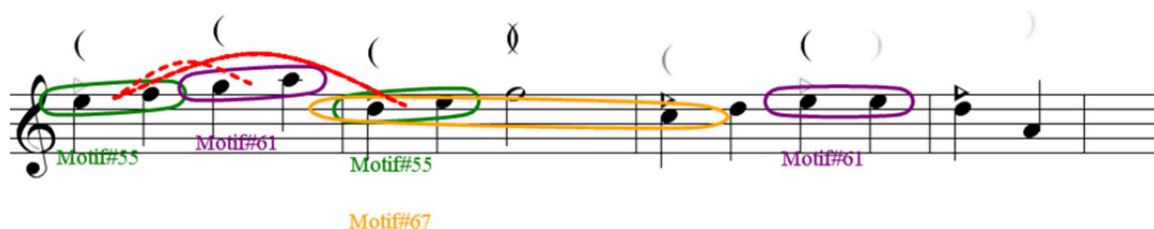


Figure 9.22: MusicatW running on “Stückchen”, note links hidden.

Several elements unique to this version of Musicat are visible in Figure 9.22. These include motifs and note perspectives.

The program looks for short repeating melodic patterns, and when it finds one, it calls it a “motif” and assigns it a unique identifier¹⁶ such as “Motif#55”. For example, in the figure, there are two groups labeled “Motif#55” and colored green. These groups share a rhythmic pattern and melodic contour: they both start on beat 1 of a measure, contain two quarter notes, and consist of a single step upwards. All of these details are part of “Motif#55”. A similar distinct motif in the figure is “Motif#61”. The two purple groups in the figure have been labeled “Motif#61”, but they are slightly different from each other in contour. Each purple group starts on beat 3 and is comprised of two quarter notes, but the first one has an ascending-step contour (G–A) while the second one has a flat contour (the note E is repeated). In this example, I looked at a separate display in the program to discover

¹⁶ Motif ID numbers, somewhat surprisingly, are always greater than 50 or so. This does not mean that there were 50 other motifs generated by the program earlier in the run (or previous runs). Rather, the motif number is set to be the same as the ID number of the node in the Slipnet that represents the motif (motif nodes in the Slipnet are discussed below). The Slipnet contains approximately 50 nodes before motifs are added; therefore, motif numbers start in the 50s.

that the standard form of Motif#61 has the flat contour, like the second purple group. However, the program has some flexibility in deciding whether a group is an instance of a motif, and in this case it assigned the (G–A) group the “Motif#61” label even though its contour was not the contour expected by the motif — the *rhythm* of the group was exactly what *was* expected, and this was good enough. In assigning both groups this label, the program has made a sort of analogy between the groups. Unfortunately, however, an explicit group link wasn’t created in this run between these two groups (a link was created to join the two green groups, however).

Behind the scenes, a motif such as “Motif#61” is represented by a “motif node” in the Slipnet. These special types of Slipnet nodes are created and added to the Slipnet by motif-creation codelets. A motif such as Motif#61 also has links to two other motif nodes: one of these describes the rhythmic motif involved, and the other describes the contour motif. Motifs based on exact pitches (instead of contours) or on sequences of “note perspectives” (see below) were also possible in the architecture¹⁷, but there were no codelets that actually created motifs of these sorts. Because they are in the Slipnet, motif nodes can become activated, just like any other Slipnet node, and activation can spread through the graph of motif nodes. An activated motif node triggers codelets to search for more instances of that motif. This mechanism worked well, in my opinion, and it should be restored in future versions of Musicat. Although motif nodes are stored in the Slipnet, I find it simpler to refer to the collection of all motif nodes as the “motivic memory”. The motivic memory can also be saved at the end of a run and used in future runs. This allows for a limited type of learning in Musicat: the motivic memory may be extended in each run to include more motif nodes,

¹⁷ This system of representing a motif as a combination of patterns along several different musical dimensions, and of seeing motifs as nodes in a conceptual network, has much in common with Olivier Lartillot’s work. See, for example, (Lartillot, 2002, 2004, 2005).

and in this way the program can in principle recognize motifs in one melody that it originally heard in another melody.

In the program's interface, groups are labeled with at most one motif, but internally, groups can be associated with multiple motifs, each one with a different strength. The display shows simply the strongest motif for each group. The collection of multiple motifs for one group is implemented using the general *perspect* mechanism that is unique to this version of the program. I borrow Ockelford's term "perspect" to describe any parameter of a note or group that has been noticed by the model (recall from Chapter 2 that the word "perspect" comes from the phrase "perceived aspect"). Each note or group has a predefined set of weighted perspects¹⁸. These include "group start", "group end", "harmonically tense", "tonic", "dominant", "leading tone", "salient", and so on. For instance, if a bass line has been given along with the melody, and if a codelet notices that a note has high harmonic tension with respect to the current bass note, then the codelet may add weight to the "harmonically tense" perspect for the note. Motif perspects are added dynamically: if a codelet notices that a group is similar to a motif stored in the motivic memory, then that codelet may add a motif perspect to the group, setting the weight of the perspect according to the degree of similarity between the group and the standard form of the motif. Perspect weights can be modified by other codelets. In some experiments I made the weights decay over time, although this generally led to too many perspects disappearing completely while they were still needed.

Three perspect types (in addition to the motif labels) are visible in the figure above. There are small right-pointing triangles that indicate perceived stressed on notes (*e.g.*, over

¹⁸ Although I use Adam Ockelford's terminology (Ockelford, 2004), Steve Larson deserves credit for the idea of having Musicat associate a list of qualities with each note. He suggested to me (personal communication, 2008) that each note in a melody "has its own personality", and that Musicat should make a list of the different qualities of each note in context.

the final D in the melody). The strength of the perspect is represented by the color of the marking, ranging from black (for a strong perspect) to light gray (for a weak perspect). The other types of perspects visible are the “group start” and “group end” perspects, represented by left and right parentheses, respectively. For example, the first note in the melody has a strong “group start” perspect, indicating that the first note sounds like a likely place for the start of group. The final E in measure 3 has a weak “group end” perspect. Perspects are not mutually exclusive: the half note F in measure 2 has both group perspect symbols, meaning that it has been heard as a likely place both for a group to start and for a group to end.

Although the perspect and motif mechanisms have much potential, they did not survive in the latest version of Musicat because they were focused on perception of very small structures (at the scale of one note or a few notes), while the latest Musicat was focused on perception of larger structures. I expect to resurrect these ideas in future incarnations of the program.

Another element implemented in MusicatW but not in other Musicats is global temperature. For example, the latest version of Musicat has only *local* temperature. I believe that global temperature makes more sense when it is applied to a small structure (such as one that can be stored entirely in working memory) than to a larger structure, such as a 16-measure melody, that cannot be perceived in its entirety at one moment. Figure 9.23, below, shows a plot generated by the program of global temperature as it changed over the course of a run.

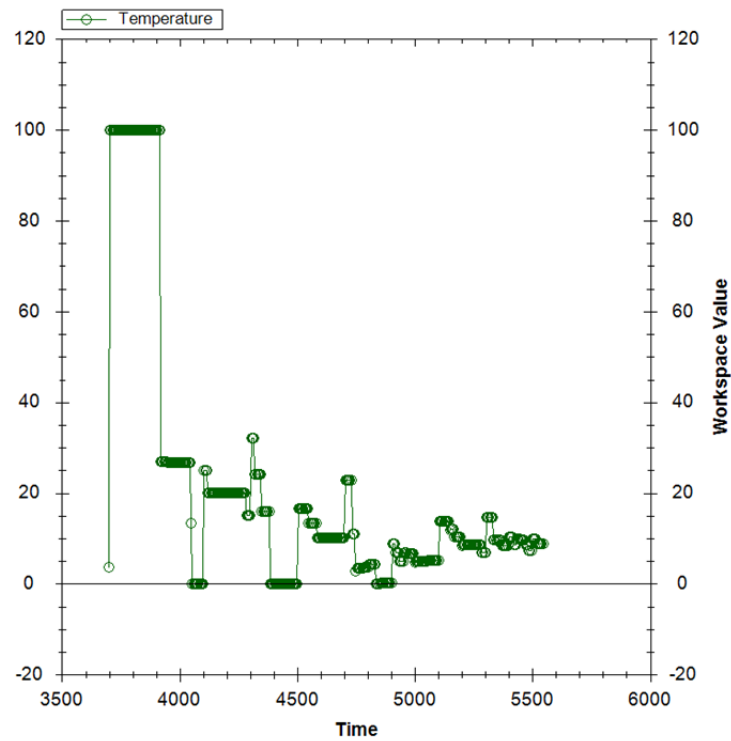


Figure 9.23: Global temperature.

In MusicatW, it is possible to add to this graph a set of lines representing the urgencies of codelets of various types. These graphs were quite useful during development, as they helped me understand how codelets were behaving during runs.

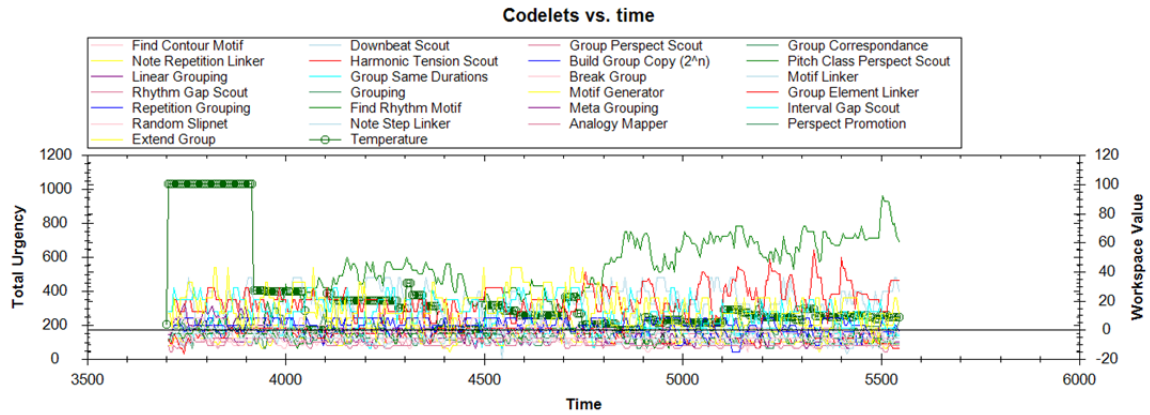


Figure 9.24: Codelet urgencies during a run.

Figure 9.24 shows the total urgency for every codelet type that was active during one run. It contains far too much informations, and so it is hard to glean much insight from it. More typically, I would include just two or three codelet types in the graph at one time, which made it much more comprehensible and hence useful to me.



Figure 9.25: Real-time codelet adjustment window.

Figure 9.25 shows a part of the user interface that I created to help during development. It is similar to the graph in Figure 9.24 in that it lists all the codelet types and displays the total urgency of each. Here, total urgency is represented by color: high total-urgency codelet types are shown in red, while low total-urgency types are shown in green. The sliders indicate the default urgency value for new codelets of each type, and can be adjusted while the program is running. Additionally, clicking on the name of a codelet type in this window will post more codelets of that type (using default parameters) to the

Coderack. I used this to experiment with the balance between different codelet urgencies and to better understand the behavior of the program as it ran.

So far I have focused on new features of MusicatW that were visible in the user interface: note links, group links, motifs, perspects, global temperature, and displays of codelet urgency. Many other features affected the program behind the scenes. The following list describes a few of the most important such features.

Key Signature and Meter

Whereas MusicatH had the goal of automatically inferring the key and meter of a melody, in MusicatW these were given to the program, so that it would be freed up to focus on other aspects of listening. Key and meter inference is an interesting aspect of listening, and one that I think this architecture would be good at modeling. However, in a real listening scenario, there are many cues relevant to key and meter that are available to a human listener but that are not part of Musicat's input, such as stress and phrasing. Thus, the listening challenge given to Musicat without the key and meter information (*e.g.*, listening to unaccented notes in perfect rhythm with no expressive timing) was perhaps harder than it should have been.

Pitch-contour Wildcards

Nodes in the motivic memory describing melody contours are very flexible, compared with how contours are represented in other versions of Musicat. A contour consists of a sequence of transitions in pitch between adjacent notes, such as “leap up — repeat pitch — step down”, just as in the latest version of Musicat. However, in MusicatW a transition can also be described with a wildcard symbol, ‘*’, indicating that any direction of

movement is acceptable. For instance, in the contour above, the “repeat pitch” element could be replaced with ‘*’, leading to the contour:

leap up — * — step down

This contour is more general than the previous one, and therefore more melody fragments could match it. A motif incorporating this contour would be a more general motif. Musicat could theoretically (that is, with additional codelets) come up with generalized contours of this sort when necessary for a particular melody. However, I explored this idea only a little because I decided to concentrate more on analogy-making in the Workspace than the behind-the-scenes analogy-making that would be involved in getting Musicat to make these more-general motifs. More importantly, the notion of trying to specify a musical motif in such a formal and mathematical manner — such as that suggested by lists of transition types and possible wildcard symbols — is not in the spirit of the fluid perception that is a goal of the FARG architecture. (It is ironic that the addition of the seemingly *flexible* wildcard character was precisely the thing that led to my realization that this sort of contour description was too *rigid*.) Olivier Lartillot’s computer models of melodic motif extraction have similar goals as MusicatW, and he reached the similar conclusion that “musical patterns actually discovered by the listeners cannot be reduced to simple mathematical objects”(Lartillot, 2005).

Context-based Scoring of Structures

Some of the challenges associated with computing the strength values for various structures in the Workspace were already discussed in Chapter 8. In MusicatW I experimented with several different scoring strategies mentioned in that chapter, but the

most effective technique I used during MusicatW's development was the idea of running statistics that I implemented in Paris in 2010: group scores were computed based on a weighted sum of a set of various criteria, and these scores were then compared with the scores of other groups scored by other codelets. The mean and variance of scores in a recent time window were used to determine how a newly-computed score stacked up against other scores. This allowed greater flexibility in designing the scoring function and choosing weights for its various features. However, I eventually stopped using this system because it was very complicated to understand the program's behavior when the strength of one structure changed rapidly based on the computed strengths of other structures.¹⁹

Tabu Search

MusicatW has many numerical parameters that I had to choose using my own intuition and experimentation. I added elements to the user interface to make it easy to modify parameters, and even to selectively disable functionality (*e.g.*, individual codelet types can be disabled, as can individual terms of structure scoring functions). This was useful in experimenting with parameters by hand. After all these parameters were exposed via the user interface, I took the additional step of making a command-line version of MusicatW that could be run by a separate program (with the user interface disabled). This external program could specify the parameter values to use in a run. The command-line version of Musicat would quickly run and then report the final state of the Workspace. Thus, runs of MusicatW could be automated, which opened up the possibility of automatically tuning the program's

¹⁹ Structure scoring based on a weighted sum of features is a very important but not well-enough understood aspect of FARG models, in my opinion. The problem of *learning* feature weights came up in my work on automatic generation of piano fingerings (Kasimi, Nichols, & Raphael, 2007) and I worked on a solution applicable to that domain (Raphael & Nichols, 2009), but for FARG models, the question of how to best approach these issues remains open.

parameters: given a set of sample melodies and desired grouping and analogy structures, the parameters could be modified automatically to try to optimize performance.

Because MusicatW is so complicated, not to mention stochastic, I considered optimization techniques that treat the function to be optimized as a “black box”. In this case, the function in question was the error obtained by performing a complete run of MusicatW on a melody and comparing the groups and analogies of the resulting Workspace with the desired groups and analogies that I specified, and then repeating the process for several runs and averaging the result, to account for MusicatW’s variability between runs. I chose the technique of “tabu search”, which does randomized hill-climbing search in high-dimensional spaces and requires no knowledge of the form of the function that is being optimized (Glover & Laguna, 1997). The “tabu” part of the name refers to a heuristic for avoiding getting stuck in local maxima: in brief, if a parameter is modified during an iteration of the algorithm, it cannot be modified again until several other parameters have been changed during later iterations.

Tabu search, once it was implemented, offered small, incremental improvements in MusicatW’s results on a small set of text melodies. However, the improvements were not very significant, and I had the strong impression that the problem was that the architecture of the program was not possible of doing significantly better, for *any* collection of parameter values — what was needed was far more substantial a change. In particular, MusicatW could not make “analogy map” structures (see below) at the time that I implemented Tabu search — those were added later.

Bar-line Thickness

I first implemented the idea of bar lines with variable thickness towards the end of the development of MusicatW (in most of the example runs below, thick bar lines are not visible because most of the runs were done with a slightly earlier version). The program preferred to create groups that would start or end just after or just before a thick bar line (with thickness dependent on group length), and to avoid crossing thick bar lines. Bar lines in this version are generated in a perfectly regular manner: the thickness of a bar line is proportional to the largest power of two that evenly divides the measure number of the preceding measure. In other words, measures 2 and 6 are followed by a bar line of a certain thickness, measures 4 and 12 are followed by thicker bar lines, measure 8 is followed by a bar line that is thicker still, and so forth. For example, recall the figure from the previous chapter, repeated here (Figure 9.26).



Figure 9.26: Bar lines of “thicknesses” following a regular pattern.

In the latest version of Musicat, bar lines *tend* to conform to this regular pattern, but there is more flexibility. In MusicatW they are *forced* to follow this pattern.

Analogy Maps

The idea of explicitly representing structured analogies (at various hierarchical levels) between groups (at various hierarchical levels) did not appear in Musicat until close to the end of development of MusicatW. In earlier versions of the program, analogy was represented in a less structured and more implicit manner, through mechanisms such as note links, group

links, and motif labels. The “analogy map” object type was added to the program to make explicit the set of relationships between two objects.

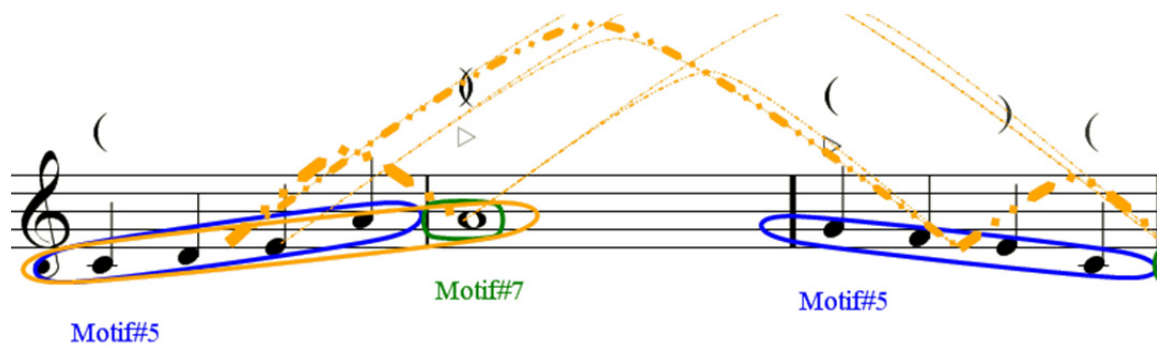


Figure 9.27: Analogy maps in MusicatW.

In MusicatW, analogy maps (or simply analogies) are displayed as thick yellow arcs with a pattern of alternating dots and dashes. Figure 9.27 gives an example in which several analogies are visible, including the admittedly strange analogy between the groups in measures 1 and 2, and a more understandable analogy between the groups in measures 1 and 3 (both groups are labeled “Motif#5”).

As was the case with bar lines, most of the examples below do not include analogies because analogies were added late in development. Analogies were much better developed in the latest version of Musicat.

Example Runs With This Version

This section presents several screenshots of MusicatW running on most of the melodies that were used as examples above for MusicatH. Two additional melodies are included at the end of this section to demonstrate some significant problems with MusicatW that motivated the major changes in the next version of the program.

The Problem of “Flailing”

In general, the examples below are primarily intended to show how much trouble MusicatW had on these short and simple melodies, even though MusicatW was a quite sophisticated program that was the result of several years of development and improvement. The program’s difficulty with these melodies illustrates both the complexity of the problem of modeling listening and the insufficiency of MusicatW’s architecture. Watching the program running was often frustrating, because it would sometimes start to make “good” groups and discover appropriate descriptions of motifs, but then it would proceed to make many “bad” groups as well. When Douglas Hofstadter was watching early versions of MusicatW running, he would often describe the program’s behavior during a run as “flailing”. During the time we were in Paris in 2010, he delighted in finding ever more colorful French words and idioms to describe the program’s ever more frustrating flailing problem, including:

- *ramer*: to row (but getting nowhere);
- *piétiner*: to be at a standstill, to stamp one’s foot;
- *galérer*: to work hard, to slave away;
- *tourner en rond*: to go around in circles;
- *pédaler dans la choucroute*: to pedal in sauerkraut

My goal was always to stop the program’s “flailing”, but this was not successful at all until the latest version (and even the latest version flails to some extent, but the situation has been much improved.)

“Stückchen”, by Schumann

Figure 9.28: “Stückchen”.

Figure 9.22, above, showed a snapshot of an earlier moment of the run depicted in Figure 9.28, but I will offer a few more comments here. The program has made several short groups of only two notes each. In general, I hear longer groups in this melody. The shorter groups make it easier for the program to find repeating patterns, such as the two instances each of Motif#55 and Motif#61, but they also reduce the amount of larger-scale analogy-making that occurs. For example, I hear measure 2 as analogous to measure 1, because both feature a stepwise ascending pattern. Measure 2 sounds like a version of measure 1 that has stopped short for a brief moment of repose on the dominant, G. MusicatW, however, “hears” only the connection between the first halves of these measures. When it does make longer groups (and meta-groups), they are sometimes hard to justify, as in the yellow meta-group that starts in measure 2, beat 1, and continues to include the downbeat of measure 3, instead of stopping after the half-note G, as I would expect. Relationships between groups are also lacking: the two red relationships in measures 1 and 2 seem good, but the rest of the melody has no relationships — not even between the two instances of Motif#61!

The group-start and group-end perspectives above the staff are interesting in this run. Most of the notes that were on a downbeat or on beat 3 were heard as likely places for a group to start, as is indicated by the ‘(’ symbols. This was a result of the program hearing beats 1 and 3 as having accents (this is built into its knowledge of common-time meter,

which has 4 beats per measure). There is a very faint, difficult-to-see group-start symbol above the A in measure 4 (beat 2); although the note A remains ungrouped, it would make some sense, to me, for a group to be heard starting here. This would be coherent with the previous note, D, being heard as the end of group. The D is also ungrouped, but it has a group-end perspective associated with it, indicated by the ‘)’ symbol, so it seems that the program was on a reasonable track in this case, even though the actual grouping structure failed to include either of these notes.

Triplet Scale

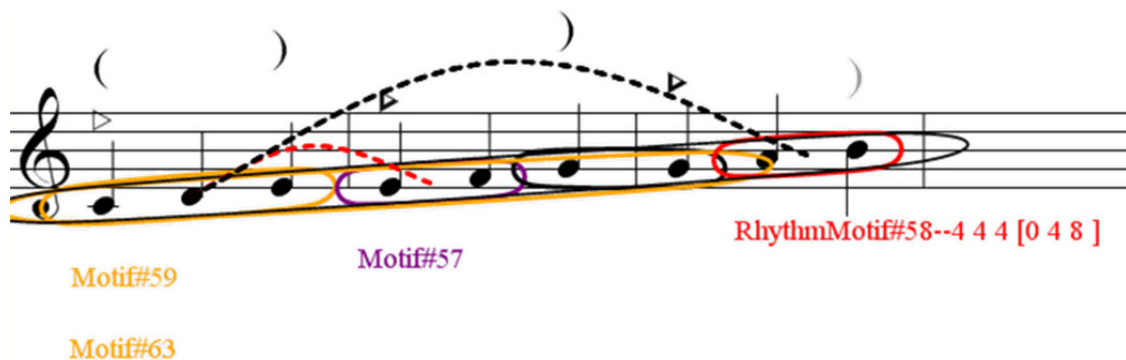


Figure 9.29: Triplet Scale.

This run on the simple “Triplet Scale” is simply terrible. It starts out well with a group of three notes in measure 1, but after that it makes three groups of two notes each: (E–F), (G–G), and (A–B). The group (G–G) is central to the problem here. According to a human listening performance, this melody is made up of three groups of three notes each, but the end of each group (except for the last one) is linked via pitch repetition to the start of the next group. For example, the group (C–D–E) ends on E and the group (E–F–G) starts on E, but unfortunately, this latter group did not form in this run. The group (G–G) was in competition with the three-note groups (E–F–G) and (G–A–B), preventing either from forming and persisting through the end of this run.

During the course of any given run, I would typically see the program creating and destroying various combinations of the expected three-note groups and rival two-note groups such as (E–E) and (G–G). This behavior exemplified the “flailing” mentioned above.

For this melody, the program would benefit greatly from understanding how the end of one group can link (via pitch repetition) to the start of the next group. These kinds of relationships are important in many other melodies, but none of the versions of Musicat (including the most recent one) pay much attention to such relationships.

Twinkle, Twinkle

For the melody “Twinkle, Twinkle, Little Star”, MusicatH recognized many of the repeated notes but did not form any useful larger-scale structures. MusicatW’s listening performances were slightly better, as I will show with a few runs below.



Figure 9.30: “Twinkle, Twinkle” (run 1).

This first run is not a particularly good run. One improvement over MusicatH, however, is that most of the notes have been incorporated in *some* group or other. Unfortunately, the groups do not make much sense, so I won’t even attempt to explain all the strange groups formed here. I will just point out that the G in measure two is the start of a large red meta-group, when it should instead be the end of the large yellow group that comes before.

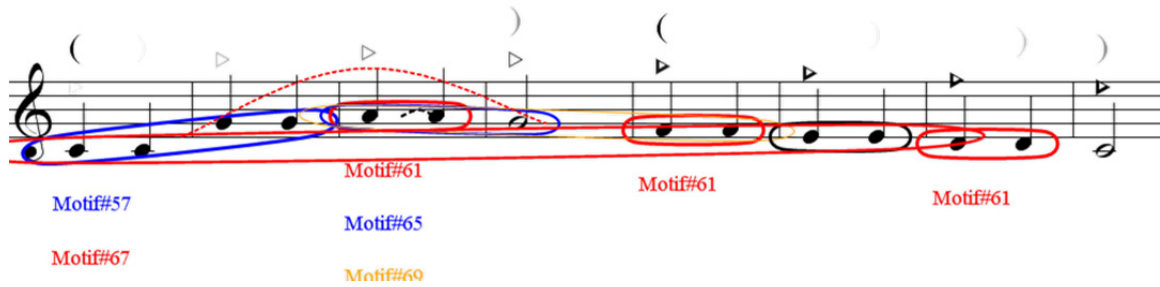


Figure 9.31: “Twinkle, Twinkle” (run 2).

In this run and the next, I told the program to interpret the music in 2/4 instead of 4/4. This run resulted in the program finding a 2-note pattern, Motif#61, which shows up in three places. Strangely, however, that motif was not perceived in measures 1, 2, or 6. Again in this run, most notes were group members, but the longer groups and meta-groups make little sense.

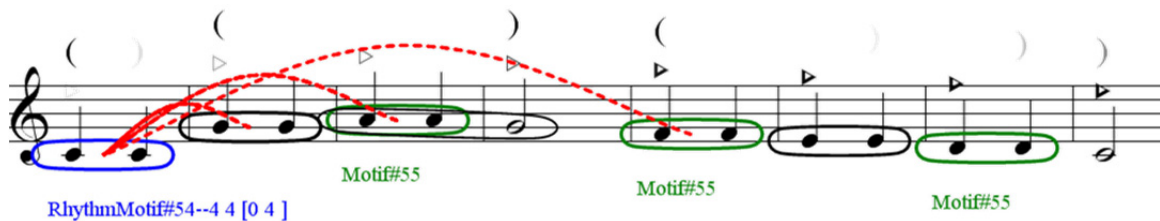


Figure 9.32: “Twinkle, Twinkle” (run 3).

This final run is very similar to the previous one. Notice the blue group at the start, labeled “RhythmMotif#54”. This indicates that the group was heard as an instance of a particular rhythmic motif, but that the pitch aspect of the group was not heard as motivic. In this example, it is very strange that the blue group is not labeled as “Motif#55”, because Motif#55 has a rhythm consisting of two quarter notes and a trivial contour consisting of a “sideways” motion (pitch repetition). Measure 1 satisfies both of these conditions, but due to the stochastic nature of the program, only its rhythmic aspect has been perceived (and indeed, three group links were made, all originating from measure 1 and linking to measures 2, 3, and 5, respectively), while its contour aspect has been ignored.

On the Street Where You Live

Recall from Chapter 7 that for the melody “On the Street Where You Live”, Musicat easily makes an analogy between measures 1–2 and measures 3–4. The program is helped tremendously by its restriction to make groups that start and end at bar lines. In MusicatW (as with MusicatH), there is no such restriction, and the program is free to make shorter groups; this freedom leads to additional complexity. In this section I first present a run of MusicatW on the first four measures of the melody (for this run and the next I use a metrically-shifted version of the melody — unfortunately, this reverses the implied accents of beats 1 and 3 in each measure, but this is a minor change). The next run and the rest of the runs in this chapter use a later version of MusicatW that can form explicit analogy maps in the Workspace.

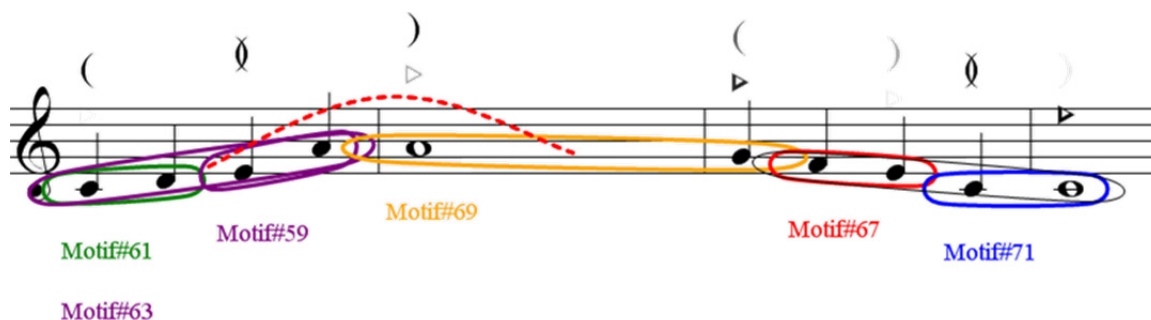


Figure 9.33: “On the Street Where You Live” (run 1).

The main problem with this run of MusicatW is closely related to the problem MusicatH had in its first run on this melody, above (Figure 9.19). In this run, MusicatW has created a yellow group starting on the whole-note A and continuing to the right to include the following quarter-note G. As before, one of the problems is that Musicat hears a strong connection between A and G because they are separated by a single step. It would make more sense for it to hear the stronger connection between the whole-note A and the quarter-note A

coming just before, because they are the same pitch! After all, the program did make the final blue group consisting of a quarter-note C followed by a whole-note C.

The yellow group is ridiculous, but aside from that, the program made reasonable groups, including a meta-group comprised of the two 2-note groups in the first measure. There is also a very weak meta-group comprised of the final two groups in the melody, in measures 3–4. If only the yellow group had not included the G, then other more reasonable groups would have been able to form and the overall grouping structure might have turned out to be understandable as a human-like way of hearing the melody! However, the group structure heard by the program lacks coherence: in this listening performance, only one relationship was heard between groups, and this relationship makes no sense: the meta-group in measure 1 has been linked to the yellow group in measures 2–3. The motif labels are also quite suspect: groups are labeled as “motifs” even though each motif only appears once in this melody: there is only one Motif#61 or Motif#67. Since each group has a different motif number, it is clear that the program has perceived no useful relationships between musical patterns; there is nothing whatsoever here that suggests that the program has heard part of the analogy $(1-2) \leftrightarrow (3-4)$ that is essential to this melody.

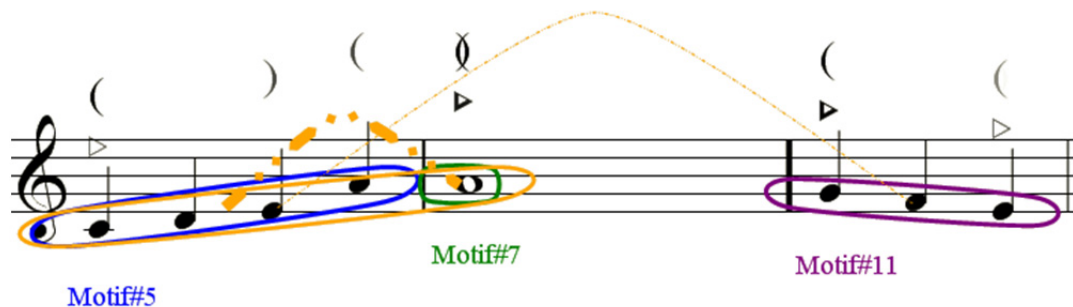


Figure 9.34: “On the Street Where You Live” (run 2, with analogies), mid-run.

In this run, I used a much improved version of MusicatW: analogy maps could be created by the program. Strong analogies are shown as thick yellow arcs with a dash-dot-dot

pattern. Tentative analogies are shown using the same color and pattern, but with very thin arcs. Another important change in this version is that I forced groups to start and end at bar lines, greatly reducing the set of possible groups that the program would consider. I stopped the run before the end of the melody to show how analogies form and grow over the course of a run.

The program has formed entire-measure-long groups in measures 1 and 2 and has combined them into an orange meta-group. The first notes of measure 3 have just been “heard” and grouped together. An analogy has been formed between the first two measures, and surprisingly, it has been perceived to be stronger than the nascent analogy between measures 1 and 3. The first analogy has a high strength because many links (not pictured) have been discovered between the notes of the measure 1 and the whole-note A. Fewer links have been discovered between measures 1 and 3 because measure 3 has just been heard.

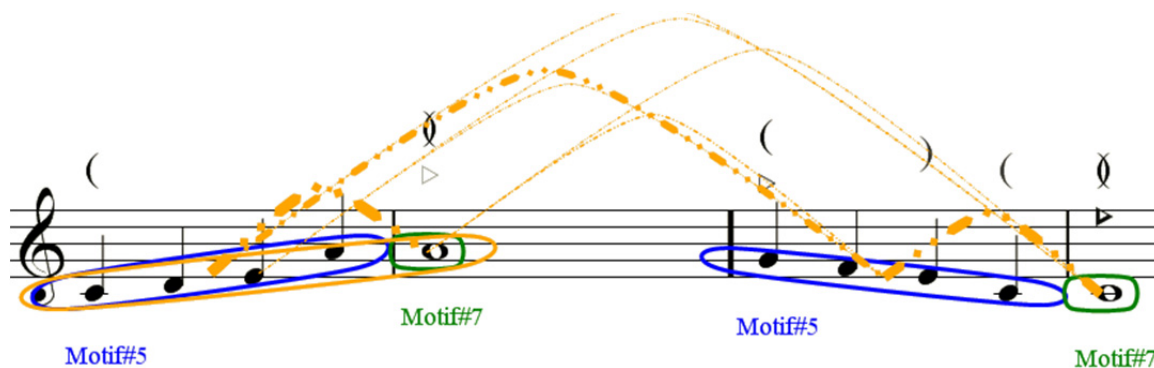


Figure 9.35: “On the Street Where You Live” (run 2, with analogies), end of run.

Figure 9.35 shows the Workspace after all four measures have been heard. Measure 3 has now been perceived as an instance of Motif#5, and the analogy between measures 1 and 3 is now much stronger than it was earlier. The final whole-note C has also been perceived as an instance of Motif#7, just as the whole-note A was earlier. A very weak analogy has been

built between measures 2 and 4; similar weak analogies have been built between several other pairs of measures.

This run, despite some problems described below, demonstrates a tremendous improvement over previous runs of MusicatW and MusicatH, which didn't have this analogy-making ability. As I consider the development of Musicat over the past several years, it is obvious to me that the introduction of these analogy maps was a pivotal moment in improving Musicat's ability to make sense of musical structure. (This version also was aided by the simplifying restriction that groups had to start and end of bar lines, but it seems that this change, which I made several months earlier, was not as important as analogy maps were.) Earlier versions of the program could make links between structures, but analogy maps provided much more. First, they provided a way to collect and coordinate many relationships between the two sides of the analogy. Second, and most crucially, they provided a source of *top-down pressure*: codelets could be posted with the goal of improving an analogy or considering the creation of groups and links suggested by the presence of the analogy. Although earlier versions of Musicat had various top-down pressures, the pressures made possible by the analogy maps provided the leverage, in many runs, for Musicat to stop flailing as it listened and to make progress in creating strong groups and links in the Workspace.

Despite all this, the program still had (and has) a long way to go. This run was quite encouraging, but for a human listener, the connection between measures 2 and 4 would be much stronger — the arc is much too thin for such an obvious relationship — and the tentative analogies between measures 1 and 2 and between measures 3 and 4 would be nonexistent. Also, this run is still missing the crucial larger-scale analogy $(1-2) \leftrightarrow (3-4)$. The program's discovery of the sub-analogy $(1) \leftrightarrow (3)$ was the big success of this run (along with a quite reasonable grouping structure).

Musical analogies that appear to us to be trivial (such as mapping the whole-note A onto the whole-note C) can pose huge problems to computer models like Musicat, because *any imaginable mapping* between any two structures might possibly be sensible in the right context. The fact that both of these notes are whole notes might be relevant in this melody, but a similar fact might be completely irrelevant in another melody (*e.g.*, consider a *cantus firmus* bass line composed entirely of whole notes). To distinguish relevant from irrelevant features is extremely hard; as designers of a perceptual architecture, we can't simply tell the program to "ignore pitch" or "ignore rhythm", because either, both, or neither of these rules might apply in a particular context.

Very Simple Melody



Figure 9.36: A very simple melody.

While working to improve MusicatW and its analogies, we composed a very simple (but not particularly musically pleasing) “melody”, which has nearly the same structure (in terms of expected groups and analogies) as the “On the Street Where You Live” melody. We realized it was essential that the program be able to generate the desired analogies in a trivial melody quickly and virtually deterministically (even though the architecture is stochastic). The melody consists of two unrelated measures followed by an exact repetition of those measures. The trivial analogies expected are between measure 1 and measure 3 and between measure 2 and measure 4, because measures 1 and 3 are identical, as are measures 2 and 4. Despite the melody’s simplicity, MusicatW still had some problems, but because of this

simplicity it was easier to figure out the sources of these problems. This simple melody served as a useful diagnostic tool.

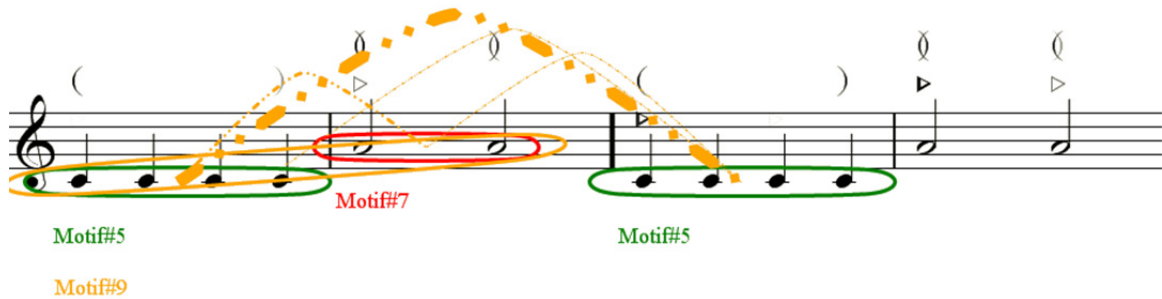


Figure 9.37: Very simple melody, during final processing.

Figure 9.37 shows Musicat after the melody had been presented, but while it was still “listening”. Just as in the “On the Street Where You Live” run above (Figure 9.35), an analogy was built between the first two measures, although a human would hear them as unrelated; luckily, Musicat’s analogy between these measures was weaker for this melody. Also as in Figure 9.35, each of the first two measures formed a group, and the resulting groups were combined to make a 2-measure meta-group, (1–2). This meta-group is sensible here (even though its constituent measures seem unrelated) because of the temporal proximity of the groups and also because of the thicker bar line between measures 2 and 3 and the exact repetition of measure 1 in measure 3. This repetition has also been recognized by Musicat, as is indicated by the thick analogy arc, representing the strong analogy (1)↔(3).

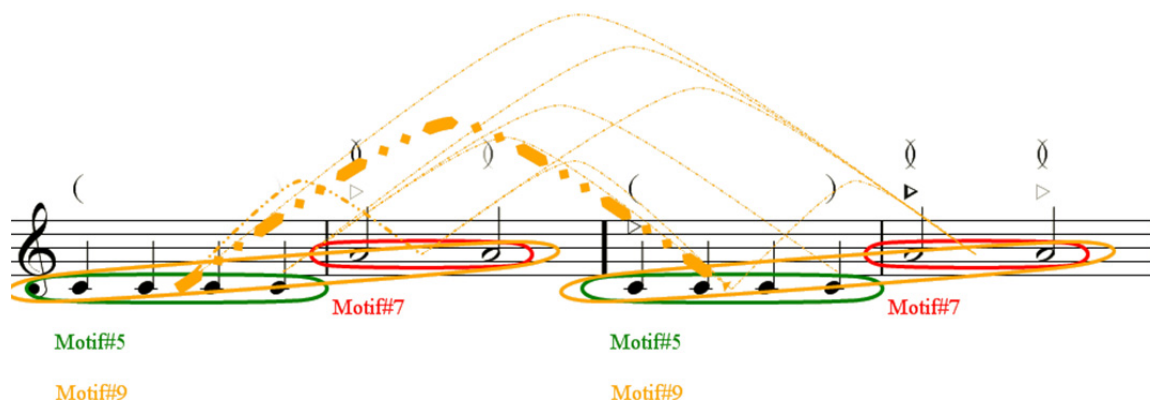


Figure 9.38: Very simple melody, end of run.

The groups and analogies formed by the end of this run are quite similar to those the program formed in the “On the Street Where you Live” run above (Figure 9.35). Each measure forms a group, and measure pairs also form meta-groups: (1–2) and (3–4). (Recall that the group (3–4) was missing in Figure 9.35.) The analogy (1)↔(3) is still strong, and many other tentative analogies have been formed of which the only one that has any significant strength is the analogy (1)↔(2), but it is still very weak, as it should be.

Unfortunately, just as in Figure 9.35, the program has missed the important analogy (2)↔(4) — a tentative analogy has been formed, but these are extremely common and so we should only be interested in analogies represented with thicker arcs. Because this analogy was missing, the program also doesn’t find the large-scale analogy (1–2)↔(3–4). This tentative analogy is also visible in the figure, but the program has not developed it.

Thus, the simple melody showed us some places where the program should be improved: we needed to ensure that the program would discover the missing analogies. *If* the program were working as desired, the (2)↔(4) analogy and the (1–2)↔(3–4) meta-analogy could be discovered in the following way: after the discovery of the (1)↔(3) analogy, a top-

down codelet might notice that the analogy crosses a relatively thick bar line (the bar line between measures 2 and 3). Then, this codelet would post other codelets to the Coderack to search for a *parallel* analogy between measures 2 and 4. Long-distance links would be built between notes in these two measures, and thanks to them the parallel analogy would easily be seen.

I added codelets to the program to make possible the scenario just described, but even so, on almost all runs, MusicatW failed to see the desired meta-analogy, $(1-2) \leftrightarrow (3-4)$. At this point in MusicatW's development, the program was quite complex, and I believe it simply spent too much time exploring other possibilities, and the top-down pressure afforded by the program's analogies was still not strong enough to overcome its tendency to "flail" too much. My experience with this simple melody, along with the program's trouble with "Sur le pont d'Avignon" (next section), soon led to a significant rewrite of the program.

Sur le pont d'Avignon

Recall the 8-measure melody of "Sur le pont d'Avignon" from Chapter 6:



Figure 9.39: Sur le pont d'Avignon.

The current version of Musicat can readily form several layers of meta-groups and make analogies of various sizes, including $(1-2) \leftrightarrow (5-6)$, $(3-4) \leftrightarrow (7-8)$, as well as the meta-analogy $(1-4) \leftrightarrow (5-8)$, mapping the entire first half of the melody onto the second half:

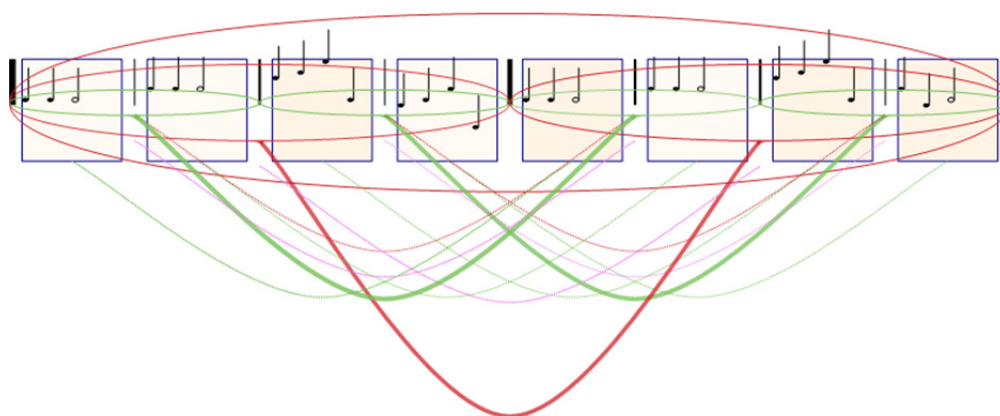


Figure 9.40: *Sur le pont d'Avignon*, result of a run using the latest Musicat (repeated from Chapter 6).

But now consider the results from a run using MusicatW. The difference is striking:

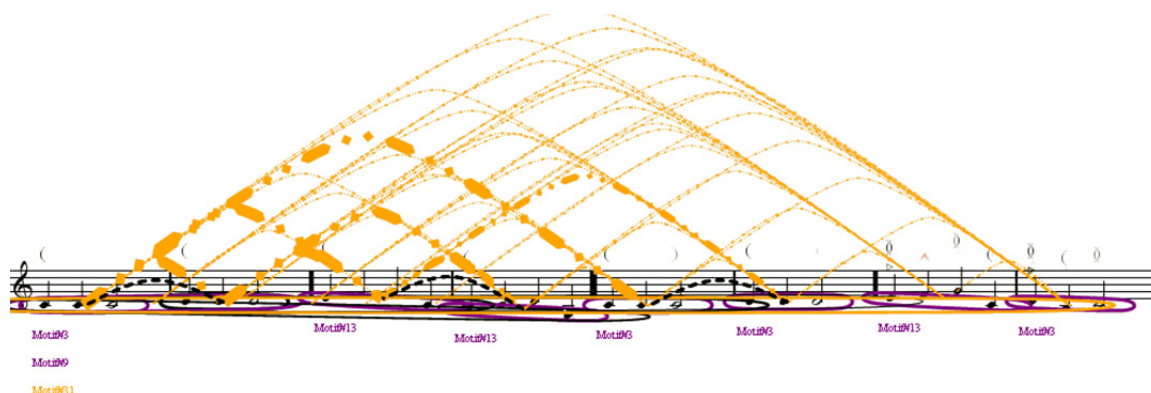


Figure 9.41: Far too many analogies in “*Sur le pont d'Avignon*” — an embarrassment of riches.

Clearly, this version of the program has completely failed to hear the structure of the melody. Instead, it has created a ridiculous number of tentative analogies: one has been formed for every single possible pair of groups! The strong analogies that have been formed seem to favor the early part of the melody, and the ending of the melody is not part of *any* strong analogy. This stands in contrast to the run of the latest version of Musicat (Figure 9.40), in which the first and second halves of the melody are placed on equal footing, as is shown by the large analogy connecting both halves.

This failure is important because the embarrassing abundance of tentative analogies that Musicat considered during the run vividly illustrates the difficult problem the program faces. In general, there are so many potential analogies between entities in the world that the combinatorial explosion is overwhelming unless one has heuristics to determine which analogies might be most pertinent. In Figure 9.41, MusicatW is looking at so many possible mappings that it simply flails, failing to perceive the most crucial features of the musical structure. The capacity to keep the focus on *what matters*, and to extract the essence — the *gist* — of a situation, is extraordinarily difficult, especially in real time. Musicat needs this ability, but it is obvious that, in this run, MusicatW was lacking that ability.

After I saw this particular “listening performance” in the spring of 2011, I was quite frustrated by the state of the program. In the fall of 2001, I therefore decided to make a radically simplified version of Musicat that would focus its attention on what I deemed to be the most important aspects of melody perception: making reasonable groups and making larger analogies based on them, and also inspiring new groups.

BINARYCAT

The early versions of Musicat included many codelets that examined the features of individual notes (note perspects) and made links between notes. Because of the run on “Sur le pont d’Avignon” in which the program failed to make any large-scale analogies (*e.g.*, mapping the first four measures onto the last four measures), I decided to try an experiment in which I removed practically all information about individual notes from the melodies given to the program. The program already had been restricted to making groups that were at least one measure long, but it still spent a lot of time “listening” to individual notes. I realized that if the program considered the music at a higher level of granularity, the patterns it had failed to perceive before should suddenly become very obvious. For example, if one

thinks of the “Sur le pont d’Avignon” melody in terms of measure-sized chunks and assigns a new letter such as **A** or **B** to each measure that has different notes from a previous measure, the melody is converted from this...



Figure 9.42: “Sur le pont d’Avignon”.

...to this:

A B C D A B C E

Surely the large-scale analogy represented in terms of these symbols, $(ABCD) \leftrightarrow (ABCE)$, is simpler to perceive than the same analogy based on individual rhythms and notes. The melody is given to Musicat in the following form, which looks considerably more complex:

QC QC HC | QD QD HD | QE QF QG QC | QB3 QC4 QD QG3 | QC4 QC HC | QD QD HD | QE QF QG QC | QD QB3 HC4

Of course, even MusicatH had the ability to make groups, and these groups should have been able to provide the chunking necessary to convert this note-level representation into a chunked representation such as “**A B C D A B C E**”. However the program seemed to be spending too much time with individual notes. I wanted to “hide” individual notes from Musicat temporarily while working on the codelets that could make analogies such as $(ABCD) \leftrightarrow (ABCE)$. I wanted to start with a blank slate and make a new type of Workspace and a new set of codelets that were aimed squarely at making larger analogies (and I planned to take lessons learned from this experiment and apply them to MusicatW).

In my first attempt to implement this idea of hiding the internal structure of each measure in order to focus the program's attention on larger analogies, I had the (naïve) inclination to simply assign a *random* bit vector to each measure with unique notes, and I made a new program called "BinaryCat" (so-named because it used vectors of 0's and 1's). This idea came to me because I had been reading about Pentti Kanerva's memory model, Sparse Distributed Memory (Kanerva, 1988), which makes elegant use of random high-dimensional binary vectors. But in this particular experiment, the use of random bit vectors was not well motivated (why not drop all notes and just use the letters **A**, **B**, **C**, and so on to represent measures?). Therefore, I worked on BinaryCat only for a few days before deciding that I should include a little more information about each measure so that the program could make useful analogies. For example, in the example in the previous paragraph, the measure labeled **D** (measure 4) looks completely different to the program than the measure labeled **E** (measure 8), but these two measures are actually quite similar.

What minimal information should be used to represent each measure without returning to a full-fledged description of every note? Douglas Hofstadter and I had already discussed how rhythmic similarity seemed more primal than pitch relationships in the initial stages of analogy formation, so I decided to give some rhythmic knowledge of the melody to this experimental program. I replaced the random bit vectors of BinaryCat with vectors representing note attack times, leading to the program RhythmCat.

RHYTHMCAT

RhythmCat is a version of Musicat that has no knowledge of pitch: input to the program is simply a list of note attack times (RhythmCat doesn't even understand rests — it just hears when notes begin). In other words, RhythmCat listens to a "flattened" version of melodies: imagine that for each melody, RhythmCat takes the original sheet music notation

and slides all the notes up or down onto one staff line. For example, the “Sur le pont d’Avignon” melody turns into the following non-pitched, percussive sequence:



Figure 9.43: “Sur le pont d’Avignon”, rhythm only.

Our decision to make a rhythm-only program may seem strange, but experiments have suggested that people are better at recognizing a melody based on its rhythm than on its pitch contour — see, for example, (Jones, 1993). If rhythmic patterns are indeed more fundamental than pitch patterns in terms of melody recognition, then it is not a large leap to suppose that rhythm is similarly fundamental for the analogy-making that the program needs to model. Pitch is also very important, but ignoring it temporarily while developing RhythmCat made it easier to focus my efforts (and the program’s efforts) on larger-scale analogy making.

BinaryCat’s code was the basis of RhythmCat. BinaryCat, however, was essentially a new program, although it utilizes small parts of MusicatW’s code, such as the code that implemented the Coderack and the main program loop for simulating the arrival of new notes and running the right number of codelets per note. I tossed out the old user interface, the old Workspace, all the old codelets, and so forth, and started anew. I created new codelets, restricted the program so that it was required to group together *measures* (not notes), a new analogy data type, new Workspace code, and so on.

The following figure shows an early run of RhythmCat on a simple rhythm.

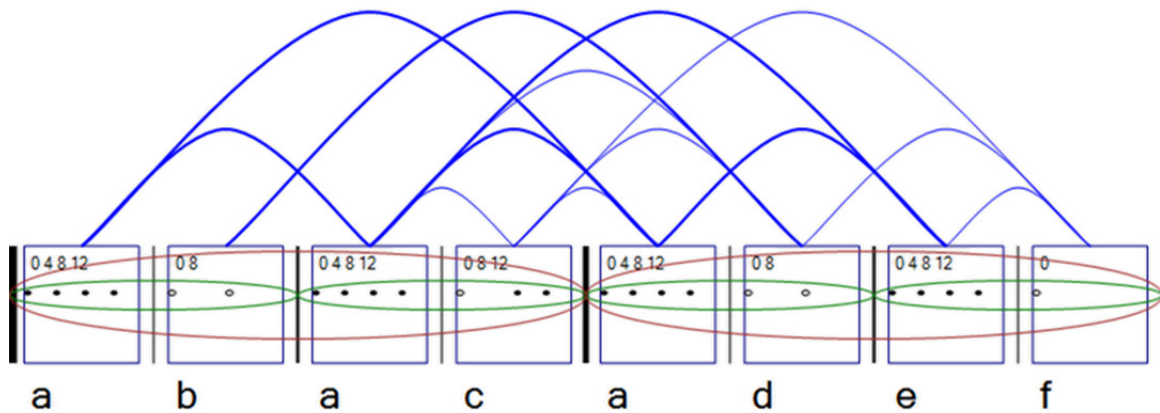


Figure 9.44: A run of an early version of RhythmCat on a simple rhythm.

This run is from a version of the program that included measure links (in blue) but didn't yet have the ability to make analogy maps. However, it did have the ability to assign labels such as “a”, “b”, or “c” to measures. For example, most of the instances of the rhythm QQQQ in the figure above are labeled with “a” (the final instance of this rhythm is labeled as “e” because the program has not yet noticed this connection). Later in its development, RhythmCat gained the ability to use the prime symbol (⌘) to mark measures as variants of other, earlier measures. The next figure shows a run from such a version, which also includes a new analogy-making ability. Also, codelet activity is indicated here by a series of yellow dots above measures where activity was taking place.

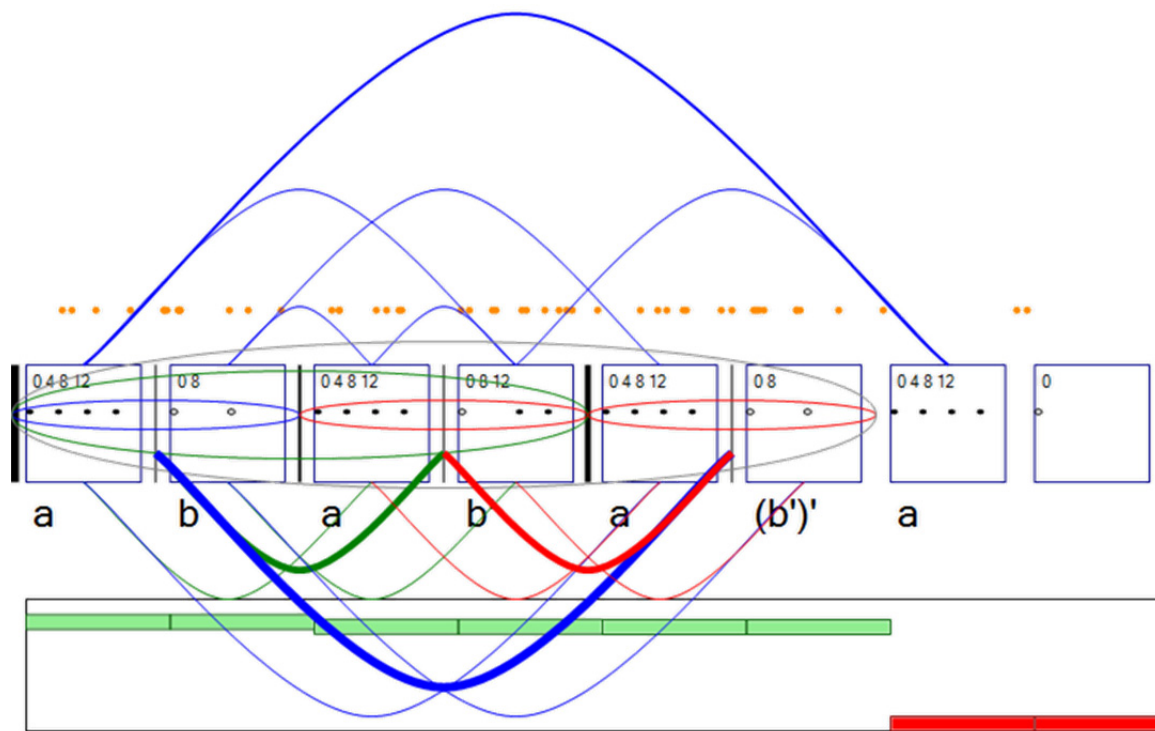


Figure 9.45: A run of a later version of RhythmCat, whose display resembles that of the latest version of Musicat.

Figure 9.45 should remind the reader of the figures in Chapters 5–7, where are based on the most recent version of Musicat. (Notice a curious thing in this run: measure 2 has been labeled as “b”. Measure 4 has been heard as a variant of measure 2, and is labeled “b”, but the “prime” symbol, unfortunately, has been covered up by a red analogy arc. Measure 6 has the same rhythm as measure 2, but it has been heard as a variant of measure 4, so it is labeled “(b’)”.)

Because it was the result of a substantial rewrite of the program code, RhythmCat has many differences from earlier versions of Musicat (although the basic FARG architecture is still the foundation of the program). The following list describes the major features of RhythmCat that distinguish it from MusicatW.

- Measures consist only of a list attack times, not pitched notes.
- There is no Slipnet.
- There is no Motivic Memory. Groups are no longer labeled with motif numbers.
- Note links do not exist.
- Rhythmic links between entire *measures* are the most elementary similarity structure in the Workspace.
- The measure (not the note) is the smallest-sized structure for the purpose of group and analogy formation.
- Group scoring uses a simple weighted linear combination (the strategy of running group score statistics was removed).
- Bar-line thicknesses are determined by codelets.
- Measure links and analogies can create special “Relationship” objects.
- Analogies always form a link between two *groups*.
- Analogies are a collection of these relationship objects in the Workspace.
- Analogies use a new scoring formula.
- Many new high-level codelets exist that add more top-down pressure.
- Temperature is computed *locally*, for a single measure or a small range of measures; there is no global temperature.
- High-temperature (low-happiness) areas get more attention from codelets.
- I implemented the add-in for Visual Studio that provides “instant” feedback after code compilation.

It may be obvious from the list above that many of these features of the program are also part of the latest version of Musicat. This is because RhythmCat rapidly evolved into the

latest version of Musicat — I did not return to the MusicatW code base. After working for several months on RhythmCat, we had seen progress but it was clear that we would have to come back to using pitch information, because there were examples in which pitch information would have helped to make certain analogies very obvious. For instance, in “Twinkle, Twinkle, Little Star”, all odd-numbered measures have the rhythm QQQQ, while all even-numbered measures have the rhythm QQH. RhythmCat notices all these repetitions of the same rhythms, but it can’t find any consistent and interesting higher-level structure without pitch. Pitch is crucial to the sense of the structure of this melody.

Because RhythmCat showed much more promise than MusicatW did in terms of making larger-scale groups and analogies, we decided to add pitch to RhythmCat, which resulted in the latest version of Musicat (that is, the version described in the rest of this thesis). In this new version of the program (which I will simply call “Musicat”), rhythm is the primary musical feature used to make analogies. Pitch was added in carefully during development, so that it would provide extra information to the program in cases like “Twinkle, Twinkle, Little Star” without overwhelming the strong clues afforded to the program by its focus on rhythm.

I have already described Musicat, but to complete the story of its evolution I will list the major new features that were added to RhythmCat in order to arrive at the current version of Musicat. The new Musicat includes the following significant new elements, among others:

- Pitch-based relationships, including contour relationships and tonal relationships;
- Inferred pitch alphabets;

- Support for upbeats (pickup notes);
- The ability to store and resurrect large and complex structures;
- A graphic display of measure-by-measure happiness values;
- Group and analogy expectations;
- A new user interface focused on showing strong groups and analogies; and
- A slider in the user interface providing a way to vary the displayed detail level.

This concludes the history of Musicat up to the current version. It is quite clear that the latest version of Musicat is a huge step forward compared with the flailing versions that came before, especially considering melodies such as “Sur le pont d’Avignon”. Many of the ideas of the earlier versions, however, may prove useful in future versions of the program. The next chapter will discuss the major lessons learned during the development of Musicat, will review the current state of the program, and will suggest priorities for the next steps in Musicat’s continuing evolution.

CHAPTER TEN

Conclusions and Future Work

In the preceding chapters I have described the architecture of Musicat, recounted the history of the program's development, and given numerous examples of the program running. In this chapter I aim to synthesize the results of this work: I review the original aims of the project, and then I describe what Musicat accomplishes in terms of modeling music-listening, as well as what it fails to accomplish. I also list what I consider to be the main contributions of this project.

Musicat is a work in progress, and I have many ideas about how to continue the development of the program, given the lessons learned so far. At the end of this chapter I list a series of suggestions for further steps, ranging from obvious and relatively simple improvements to quite speculative ideas.

AIMS OF THE PROJECT

The ambitious long-term goal I have in mind for Musicat is to make the program simulate human listening for a wide range of musical pieces of varying complexity and styles, including not only monophonic melodies but also works for multiple voices and instruments. The current version of the program has the restricted but still incredibly difficult

goal of “listening” to monophonic, tonal, Western, song-like melodies. Not only is music-listening incredibly complex, it is also an idiosyncratic and essentially human activity, making it hard to evaluate a model of it. How should one judge the accomplishments of Musicat?

Although the long-term goal is to simulate human listening, a more central role of cognitive models such as this one is to examine possible mechanisms that form the basis of human thought. In the case of Musicat, the modeling effort hopefully provides some insight into certain aspects of human music-listening. Also, because it is a computer model, Musicat makes concrete some heretofore theoretical aspects of music cognition (such as the importance of analogies in melodic structure).

What Musicat Does Well

The following list of “things Musicat does well” has an intentionally optimistic flavor: I describe some of the best things about the Musicat program, even though I freely admit that in every one of these instances it still has an unbelievably long way to go before it can claim to listen in more or less the same way that humans do. It is easy to see that the latest version of Musicat does much better than the earlier versions (as described in Chapter 9), but it is just as obvious that much more work is needed.

MUSICAT BEHAVES DIFFERENTLY ON “BAD” MELODIES

In Chapter 5, I gave several examples of “bad” melodies. These were “bad” in the sense that they didn’t have any large-scale structure: individual notes or measures were quite random. I ran Musicat on those melodies after finishing the current version of the program, and I had some concerns that it might generate just as many groups and analogies for these melodies as it did for well-structured, “good” melodies. Musicat did make more groups than I had hoped, and they were usually based on its default preference to make a perfectly regular

hierarchy of groups with powers-of-2 lengths. Encouragingly, however, it rated many of these groups as quite weak. Moreover, it spotted far fewer analogies in these melodies than it did in “good” melodies, where were covered in the following chapters.

In general, Musicat seems to find less structure in “bad” melodies than in “good” melodies. While this may seem unimpressive, it is an important sign that the program is “hearing” structure in a reasonable way. It is easy to imagine a program that would take “garbage” melodies and would generate complex structures in its listening performances, but happily, this does not seem to characterize Musicat’s behavior, at least not for these examples. However, as discussed below, Musicat’s understanding of tonal implications in music is very limited, so it does indeed respond in essentially the same way to a melody that is “bad” in terms of pitch structure as it would to a normal melody. If I were to insert random “errors” in a melody, say, by adding accidentals in tonally inappropriate places, the program would not notice the mistakes. Despite this blatant weakness in the pitch domain, the program’s different types of performances on “good” versus “bad” melodies is encouraging.

MUSICAT GENERATES HIERARCHICAL GROUPING STRUCTURES

Music theories such as GTTM have emphasized the importance of hierarchical grouping structure in music. The analytical techniques of GTTM, in particular, result in tree structures that describe musical works in terms of hierarchically-nested groups of various durations, ranging from groups of just a single note or two all the way to groups spanning the entire duration of a piece. To the best of my knowledge, few computational models generate hierarchical structures of this nature (with some exceptions, such as explicit attempts to generate GTTM analyses by computer (Hirata et al., 2007)). Recent computer modeling more typically aims to generate single-level segmentations; that is, music is partitioned in time but groups are not nested within other groups.

Musicat generates hierarchical grouping structures as it “listens” to a melody. Such structures are important not only because they appear in music notation (beams and phrases, for instance) and in music theory, but also because they are crucial in music cognition: limitations of human memory necessitate chunking of musical structures. Chunking allows people to make sense of musical structures of widely varying durations. More specifically, chunking allows people to understand and remember musical segments that, at the level of notes, are far too large and complex to represent at one time in working memory.

MUSICAT LISTENS IN REAL TIME WITH LIMITED WORKING MEMORY

Not only does Musicat *generate* hierarchical grouping structures, but it does so *in real time*. If we were not interested in cognitive modeling, then there are obvious ideas from computer science that might be used to partition a melody into segments if the entire melody were given to an algorithm as a whole — see, for example, my own work on audio partitioning (Nichols & Raphael, 2006) — and one can imagine using a similar algorithm to create a hierarchical grouping structure. However, this type of algorithm is not cognitively plausible, obviously, because it requires the whole melody to be analyzed at once. (Furthermore, without a mechanism for analogy-making, I doubt that such an idea would work well in any case.) Human music-listening involves making hierarchical grouping structures as a piece is heard, despite the limited capacity of working memory.

Musicat generates hierarchical structures in real time without modifying grouping decisions made far in the past. Small changes to recent groups are allowed (and indeed are necessary, to provide Musicat with its fluidity), but as groups fade into the past, they become fixed and are eventually treated as inviolate. These chunked memory items from the past retain the ability to be referred to by analogies or incorporated into larger groups. Thus, even as time flows, with new notes being “heard” and older notes fading into the past, Musicat can

form ever larger structures whose constituent elements would be too large to store in working memory. It is remarkable that people can make sense of musical works whose durations are orders of magnitude greater than that of the phonological loop, and with number of notes orders of magnitude greater than Miller's magic number seven (plus or minus two); it is chunking that makes it possible. Musicat does not model the *specifics* of human working-memory limitations in great detail, but it does give a concrete implementation of mechanisms that can build large hierarchical structures despite the sorts of constraints imposed by human memory.

I designed Musicat to model real-time listening, but "real time" was meant to be simulated: Musicat was given notes one at a time at a constant rate, but I did not worry about the specific details of computation time or processor speed. I would have been happy if the model could "listen" to a melody even if the melody had to be presented at an extremely slow tempo. However, because Musicat is constrained to use a small amount of working memory and considers chunked groups from the past instead of "hearing" all the notes of a melody at once, it is quite efficient. Indeed, when the computationally intensive graphics-drawing code in the user interface is disabled, Musicat can "listen" much faster than real time (*e.g.*, it can simulate listening to a 20-second long melody in a fraction of a second). This speed is possible because a fixed number of codelets are run for each beat, and no codelets have loops that will run for a long time because the size of any loops has an upper bound determined by the small number of items in working memory along with a small number of accessible groups in the past. Every codelet runs quickly, and the time complexity of the program for each incoming note is constant — $O(1)$ in computer science's "big- O " notation — because of the small upper bound on working memory size. For long melodies, the number of groups in the past does grow, but access to these could be, in principle, constrained so as to ensure constant time complexity per note even for very long melodies.

This is necessary for a cognitively-plausible model of human listening — obviously, people do not require more computation time per note as a melody gets longer. Musicat's design allows quite naturally for scaling-up to long melodies: melody-length is a non-issue, thanks to the constraints imposed by real-time listening.

MUSICAT NOTICES DIFFERENT THINGS ON DIFFERENT RUNS

Since Musicat is a stochastic program, each run is different. While randomness is a core component of the FARG architecture, it has unique implications for perception in the temporal domain of music. First, in each run, Musicat focuses its perceptual attention on slightly different structures. While this is also the case in other FARG programs such as Copycat, which might pay more attention to certain aspects of letters in one run than in another, or Cappyblanca, whose simulated random eye saccades cause the program to pay attention to different squares on a chess board in different runs, Musicat has a fixed amount of time to allocate to structure perception during each run. Copycat or Cappyblanca can run for quite a while on a given problem, and the programs themselves regulate (at least to some extent) the amount of processing time that is used per problem. Musicat, however, has its processing-time determined by the rate at which new musical notes are heard. This relatively limited amount of processing time results in runs that are very diverse, because of Musicat's different focus of attention for different runs. A second implication of Musicat's randomness combined with real-time listening is that the perceptual choices Musicat makes early in a run can have significant consequences later in the same run. For example, an analogy can only be formed between measures 1–2 and measures 9–10 for some melody if the group (1–2) was formed while Musicat was listening to the first several measures; after enough musical time passes, the program can no longer go back and modify the grouping structure of measures heard earlier.

Musicat's ability to focus on different parts of a melody on different runs reminds me of my own listening experience: I know that I pay attention to different things during different sessions of listening to the same piece. Sometimes this is more consciously controlled than is the case for Musicat's listening: for example, I might decide to pay attention to an inner voice or a bass line while listening to polyphonic music. Or, in the case of monophonic melody, I might decide to focus my attention on a particular musical motif or even an individual pitch. Musicat does not control its own attention in this way (perhaps a future meta-Musicat could do such a thing), but it does notice different things on different runs, and this also happens with my listening: I might listen to the same recording of a song dozens of times, but, quite haphazardly, I might notice different aspects of the music during different listenings. I suspect that this is one of many things that make some melodies so much more amenable to repeated hearing than one might expect: rather than getting bored from over-familiarity, listeners can discover new things on new occasions.

MUSICAT FORMS ANALOGIES

The most unique thing about Musicat, compared with other models of music cognition, is that it makes analogies between musical structures. Musicat sometimes misses analogies that seem obvious to people, and other times it makes analogies that seem non-justifiable, but on the whole it makes many analogies that seem reasonable to me. For instance, in "Row, Row, Row Your Boat", Musicat generated the chain of analogies $(1-2) \leftrightarrow (3-4) \leftrightarrow (5-6)$, which is a sophisticated and human-like way of hearing the first six measures: measures 3-4 are heard as a development of measures 1-2, and likewise measures 5-6 are heard as a development of measures 3-4. The program also makes larger-scale analogies, such as $(5-8) \leftrightarrow (9-12)$ or the even-larger $(1-8) \leftrightarrow (9-16)$ in "On the Street

Where You Live”, thus illustrating that it is able, to some degree, to make sense of musical structure.

Analogy-making is a crucial component of the human experience of listening to music. Even though Musicat’s ability to form analogies is terribly limited compared with a human’s, the mere fact that it is able to “hear” analogies at all is a success. Additionally, Musicat’s ability to form *multiple* analogies within a single melody gives it the potential for hearing melodies in a rather sophisticated way that is evocative of the complexity of human listening. Much of the meaning of a musical work (especially in the case of non-programmatic “absolute” music) derives from the very specific and unique internal vocabulary of musical ideas that exists only within that particular work. Except for trivial cases of exact repetition, such a vocabulary is recognizable thanks to analogies formed between elements of the vocabulary. Analogy-making, then, gives Musicat a preliminary, but promising, way of understanding certain aspects of musical semantics.

What Musicat Does Badly

Admittedly, there are many aspects of musical listening that Musicat does not model well, or at all. Music is highly complex, and Musicat just scratches the surface of music-listening, so the list of deficiencies in this section could be made arbitrarily long. Therefore, I focus on things that Musicat currently does badly but that I believe it *should* be able to do with some improvements but without any radical changes to the current architecture. Musicat doesn’t attempt to model the emotional experience of music-listening, for instance, so that topic — among countless others — is not addressed here.

MUSICAT FORMS GROUPS AT MEASURE BOUNDARIES ONLY

As I have mentioned many times in previous chapters, in the latest version of Musicat (in contrast with the earlier versions, MusicatH and MusicatW), groups must start or end at measure boundaries (in the case of melodies with pickup notes, this is true if we consider the measure boundaries to be shifted to the left by the total duration of the pickup notes). As a result of this simplification, Musicat became much better at forming large groups and analogies, but this constraint is too restrictive. After all, salient groups (from a human point of view) can occur at the sub-measure level. One particularly important class of examples of this problem is *melodic sequences*: if a sequence is built on a half-measure-long motif, for instance, Musicat is not able to recognize the sequence because it cannot build analogies among such short passages.

Additionally, groups often have a duration that is equivalent to a non-integral number of measures. For instance it is common for a single note to serve a dual function as the end of one group and the start of a new group, as was discussed in Chapter 2 in the section about GTTM. The current restriction prevents this sort of group overlap.

MUSICAT MISSES TOO MANY SIMPLE THINGS

There are many musical relationships or places that have group boundaries that are exceedingly obvious to a human yet seem to be imperceptible to Musicat. For example, in “Younger than Springtime”, Musicat usually failed to make the analogy (1–2)↔(3–4), linking together the phrases for the lyrics “Younger than springtime are you” and “softer than starlight are you”. Even without the lyrics, the analogy between the phrases is blatantly obvious to a human listener. The phrases share such a distinctive rhythm and pitch contour that the analogy seems to scream out at the listener. The program did make the analogy sometimes, which is a relief to me, but it should have made it *every* time. In general, Musicat

does not notice these sorts of extremely salient relationships nearly as easily as it should; it seems that it works too hard trying to form more complicated groups and analogies, at the expense of simple structures.

MUSICAT MAKES ERRORS IN GROUPING AND BAR-LINE THICKNESSES

Musicat makes two important types of errors in grouping: determining starting and ending points of groups and determining how to combine two groups into a meta-group.

Musicat relies on its perception of bar-line thicknesses (in other words, its perception of hypermetric hierarchy) to make many decisions about grouping. Sometimes it is not flexible enough and doesn't allow a group to cross a pre-established bar line even though that would result in a better group (and as a result, it might result in changing the bar line thicknesses — bar-line thicknesses and groups influence each other in Musicat). For example, in “Sun and Moon”, a natural group boundary is *after* the whole note on the word “sky” in measure 9, but Musicat hears a thick bar line *before* measure 9 and always creates groups that end before the whole note. On the other hand, sometimes Musicat is too flexible, and allows groups (especially large groups) to straddle thick bar lines when doing so is inappropriate. Often the program correctly hears the bar line between measures 8 and 9, for example, as being quite thick — after all, many melodies have a strong cadence on measure 8 — but then it will ignore this clue and form a group such as (7–10).

Musicat also has a tendency to make meta-groups whose constituent groups are of mismatched size. For example, sometimes an 8-measure group such as (1–8) and a small group such as (9–10) will be used to make an out-of-balance meta-group (1–10), without any obvious justification for combining groups of such different durations. Musicat penalizes meta-groups for such a size mismatch, but with the program's current parameters, there are

often other positive factors that override the penalty, and allow the awkward group to be formed.

MUSICAT MAKES TOO MANY ANALOGIES

Although Musicat sometimes fails to make some nearly trivial analogies, on the whole it errs on the side of making too many analogies. For example, recall the second run of “On the Street Where You Live”, in which Musicat made 12 analogies in the first 16 measures, all between various 2-measure groups. Most of these analogies were defensible, and some were stronger than others, but this abundance of analogies doesn’t seem to capture the most important things that I hear when I listen to the melody. One problem, I believe, is that this version of Musicat makes analogies only between elements in the Workspace. In contrast, MusicatW could create motif nodes in its Slipnet (*i.e.*, in motivic memory), and then analogies could be formed between groups in the Workspace and groups stored in motivic memory. Many of the analogies between the groups in the Workspace that seem to be extraneous in this version might be better represented as analogies to nodes in the Slipnet. In other words, a revised version of Musicat form *categories* in the Slipnet to represent motifs, and any time it would hear a new instance of a motif it would be reminded of the category. Categorization is analogy-making; reminding is analogy-making. These processes would complement the current analogy-making that only takes place among elements in the Workspace.

Another issue leading to the overabundance of analogies is that Musicat is simply too flexible in forming correspondences between two structures. Often, such flexibility is exactly what is necessary for high-quality analogy-making, but in some cases the program seems to be finding analogies based on far-fetched connections between structures. Analogy-making is,

of course, much subtler than what Musicat can currently do with its analogy objects and analogy-finding codelets.

MUSICAT MAKES TOO FEW ANALOGIES

Even though I have just finished describing how Musicat makes too *many* analogies, in another sense it makes too *few* of them. Specifically, recall Chapter 4, in which I described how people make analogies across a huge range of levels, from the note level (or even below), up to the level of entire pieces (and even above — Shostakovich’s 24 Preludes and Fugues were composed by analogy with Bach’s 24 Preludes and Fugues in the *Well-Tempered Clavier*). Musicat makes analogies in which each side of the analogy is at least one measure long, but as discussed above, many more analogies can be found at the sub-measure level. And even at the level Musicat works at, it does sometimes miss analogies that would be obvious to humans. In sum, Musicat makes too many analogies overall at its usual level of analysis, but too few analogies overall when considering smaller levels as well.

MUSICAT IGNORES MANY CRUCIAL ASPECTS OF PITCH

Musicat’s development followed a curious trajectory: I initially gave the program melodies in which all notes had the same duration (*i.e.*, they were all quarter notes), and in which pitch was the most important component of each note. By contrast, in RhythmCat (and also, to a great extent, the latest version of Musicat), rhythm is king and pitch is a treated as a second-class citizen. This reversal was necessary to make Musicat recognize the more primal similarity of rhythmic patterns, but it makes the program much less sensitive to pitch than desired. For example, people are quite good at detecting “errors” in pitch in simple tonal contexts — a single stray F# in a C major folksong that is unknown to the listener will

not go unnoticed. Musicat, however, will be only slightly affected by strange accidentals inserted into a piece, especially if the piece has a repetitive rhythmic structure.

If Musicat listened to polyphonic melodies or to melodies that were paired with explicit harmonic progressions (as opposed to the harmonies that it infers behind-the-scenes), there are obvious techniques that could give Musicat more sensitivity to pitch. For example, the degree of dissonance for each melody tone could be calculated with respect to the harmony using existing techniques such as those in *Tonal Pitch Space* (Lerdahl, 2001). However, there is much more to pitch than consonance or dissonance, and even in the monophonic case (Larson, 2012), pitch movement has subtle and intricate ramifications. Overall, pitch is much more crucial to music listening than is implied by Musicat's current design.

MUSICAT DOESN'T GENERATE NOTE-LEVEL EXPECTATIONS

My original plans for Musicat included having the program generate real-time expectations about which notes might come next. I toyed with this idea in MusicatH, and even in MusicatW for a while, but the current version does not make this sort of explicit expectation. Instead of expectations involving specific notes, it makes *structural* expectations, which are simply expectations that a strong group structure will repeat as soon as the group reaches closure. Some amount of melodic repetition is implied here, but at the note-to-note level, the program doesn't care which way the melody might move. This is an important deficiency (although I believe it would be relatively easy to implement note-level expectation at this point), because a program with expectations for notes also has the ability to be "surprised" or "satisfied" with notes as it hears them. Real-time denial of or satisfaction with expectations is a crucial component of the affective listening experience.

MUSICAT FLAILS

One of the biggest problems for Musicat is what Douglas Hofstadter has called “flailing”: the program sometimes creates excellent groups and analogies, only to destroy them moments later in favor of new and often terrible structures. The program often alternates between several competing grouping structures during a single run, and fails to settle down on what, to a human listener, is obviously the strongest choice. Although Musicat computes strengths for various structures and should be able to tell which among various rival grouping structures is the best, it keeps destroying strong structures. Although the program seems to flail less now than in previous versions, the problem is far from solved at this point. It is worth considering the issue for a moment so that this program (and others) might avoid the problem in the future.

Why does Musicat flail? A more precise question is: why is flailing a problem with Musicat, and not of FARG-architecture programs in general? After all, the notion of a subterranean “fight” between various competing pressures and between rival structures in the Workspace is fundamental to this type of architecture. At the *subcognitive* level, some degree of “flailing” is expected and even (when temperature is high) encouraged. But why don’t these many tiny “fights” result in an unstable, chaotic Workspace that never settles down to a good representation? Why doesn’t micro-flailing translate into macro-flailing?

In brief, the answer is that although individual codelet actions do indeed act in a rather frenzied, chaotic-seemed way, creating and destroying structures with great speed, a stable structure does tend to emerge as a result of all these actions (at least in well-behaved FARG programs!). Strong structures and regular behavior can result from a seemingly chaotic foundation when there are statistical regularities that bias the system in a particular direction. Take a gas, for instance. The individual motions and interactions of particles of a gas are far too numerous and chaotic to measure, but the aggregate behavior is not chaotic at all: the

temperature and pressure of a quantity of gas contained in a fixed volume are quite simple properties to understand and to measure. So what is it about Musicat that makes its aggregate behavior more chaotic than desired?

I believe that there are two main problems for Musicat: it does not pay enough attention to extremely obvious and even superficial cues, and it does not make use of Workspace temperature and simulated annealing in quite the same way that earlier programs such as Copycat did.

Regarding the first problem, there are certain cases where groups and analogies are exceedingly obvious, and the program should simply jump to a conclusion in each of these cases without spending much time considering other possibilities. Domain-specific knowledge (embodied in codelets) may be particularly helpful in this respect. For example, if a melody moves downwards through a scale in quarter notes and then ends on the tonic in a whole note, the program's default assumption should be to hear the scale and the whole-note tonic as constituting a group. Extremely strong pressure should be required to change that default grouping. The current version of Musicat, however, does not see obvious groups of this and other types quickly enough, and even when it does, it is far too eager to entertain alternate possibilities.

The second problem, having to do with temperature, is related to the previous issue. In Copycat and related programs, alternate possibilities were paid much greater attention when temperature was high. Musicat does use a type of temperature, but, as discussed in Chapter 8, it is computed locally for measures, groups, and the current working memory area, rather than globally, as in Copycat's Workspace. Global temperature in Copycat had an important effect on competitions between rival groups: high temperature made the program more likely to consider alternate groups, whereas low temperature resulted in increased stability for strong structures. The average temperature (or happiness) of the structures in

Musicat's working memory is used in an analogous way to global temperature in Copycat, but perhaps temperature is less useful in this case because the working memory (the past four to eight measures) is quite large and is the host to *many* groups and analogies. Temperature affects competitions between rival groups, but perhaps a more local notion of temperature would be more effective for Musicat. Additionally, simulated annealing in Copycat forced the temperature to descend over time. In Musicat there is no simulated annealing, and although the strengthening of structures as they fade away into the past fills a somewhat analogous role, perhaps the lack of simulated annealing is another contributor to the problem of flailing.

MUSICAT DESTROYS LARGE STRUCTURES

This issue might well be considered a version of the flailing problem, but I address it separately. Musicat sometimes creates quite strong and large structures, including groups, analogies, and expectations, but soon thereafter destroys them and lets them go forever. This sort of problem was addressed in the Tabletop program, and as I explained in Chapter 8, I implemented a solution based on that in Tabletop: large strong structures are retained in memory after they are destroyed and they have a chance to be brought back to the Workspace quickly, bypassing the need to recreate all their individual components. However, this idea didn't work very well at all, perhaps because of technical issues related to the temporal nature of the domain. This problem may turn out merely to be a subtle bug rather than a deep issue. I have no way of knowing, at this point.

In the case of expectations, however, there is a special problem to be solved that is somewhat related to the flailing issue. Whereas most of the listening architecture is quite dynamic, with groups and analogies being rapidly created and destroyed until the system (hopefully) stabilizes on a strong structure, expectations sometimes need to be more

persistent, in the sense that they need to remain for quite some time without being destroyed. For example, consider a situation in which the program has heard eight measures of a melody and has formed an expectation for an analogous 8-measure structure to follow. In the current version of the program, a large-scale expectation like this can readily be formed, but to be useful it needs to remain active in the Workspace for the full duration of those eight measures. Unfortunately, even when a very reasonable 8-measure expectation has formed, sometimes Musicat will temporarily form a group of a length that is incompatible with the expectation (such as a 3-measure group), and even if the temporary group turns out to have low strength and is quickly destroyed, its creation may result in destruction of the 8-measure expectation. Thus, for large-scale expectations, Musicat needs a way to balance its need for flexibility in trying out many possible group structures with the need for expectations that can persist long enough to have a tangible top-down effect on perception and can guide the formation of future expected groups.

MUSICAT GETS CONFUSED BY LONG MELODIES

As was demonstrated with the “Tennessee Waltz” melody (Chapter 7), Musicat gets confused when a melody is too long (say, longer than 16 measures). The program makes groups that span long portions of the melody in strange places without much apparent justification; moreover, meta-groups form and create alien-looking patterns, and analogies often link groups of disparate sizes together. Part of the problem is simply an abundance of options for the program’s codelets to consider: in longer melodies, there are more levels of grouping hierarchy present, more options for making meta-groups, and more opportunities for the program to mistakenly make analogies between groups of quite disparate sizes. Most importantly, there is simply more material to consider. Although the size of working memory is limited, the program makes analogies to groups that occurred earlier, and it probably does

so too often. More-limited access to older groups, along with an implementation of motivic memory, should help improve the program's behavior on longer melodies. Because Musicat already can handle melodies that are 8 or even 16 measures long and because the focus of most codelets is on the most recent several measures only, I think that many of the problems associated with scaling up to "long" melodies have been handled already, and I expect that the remaining associated problems are manageable.

Contributions of Musicat

The Musicat program and its listening performances may be the most tangible product of this research, but the problems encountered along the way, the design choices made to address them, and even the problems that the program still has are also useful results. In this section I list what I consider the most important knowledge I can impart based on my work on Musicat. This includes lessons learned in designing the program, new ideas that extend the FARG architecture, and contributions to the modeling of music cognition.

LESSONS LEARNED

The following lessons that I learned while working on Musicat apply to FARG models in general. The final item, however, pertains specifically to melody.

Aim for Deep Superficiality

Perception is incredibly complex and subtle. However, there are some simple perceptual cues that are quite salient and important, and a flexible model should rapidly and reliably exploit them, even if they seem superficial. For instance, Musicat should easily notice and pay attention to such things as exactly repeated notes or patterns, very abrupt changes of

rhythm, extremely high or low notes, passages that form scales or arpeggios, tonics and dominants, and so forth. This strategy is especially crucial in a real-time domain because the time available for perception is limited. Unlike Copycat, which had ample time to consider many alternate ideas for each analogy problem, Musicat must rapidly and accurately discover the most important things. In so doing, the program can, paradoxically, listen in a more sensitive way than it would otherwise, because by very rapidly perceiving some things thanks to knowing how to take advantage of surface-level cues, it frees up more time for deeper, subtler, and more meaningful analogy-making. Douglas Hofstadter refers to this strategy as “deep superficiality”.

Don't Downplay Top-down Pressures

A common issue we encountered during Musicat's development is that the program would get hung up on details of low-level perception, flailing about in making small-scale structures such as groups of just a few notes or a few measures. However, top-down pressure can greatly help to organize and prioritize lower-level perceptual processes. Many of the biggest advances in the quality of Musicat's listening performances came when I added new types of top-down pressure. For example, creating codelets that would take an existing analogy and look for a *parallel* analogy was particularly effective. The *high-level* search for the second analogy would be undertaken by additional codelets that would bias the program's *low-level* perception so that it would tend to look for the key groups and relationships in the most natural, fairly predictable places.

Rapid Feedback during Implementation is Essential

All runs of a stochastic program such as Musicat are different, and each run takes quite a while to complete when the visual display of the program's Workspace is enabled.

Therefore, understanding the effect of recent changes to a codelet or to parameter settings by simply running and rerunning the program requires a significant amount of time: I might need to watch the program running three, four, or even ten times to understand the effect of such changes. Luckily, I was able to streamline this process and to understand the effects of changes much more easily once I had designed a rapid-feedback tool (the Visual Studio add-in described in Chapter 8) that allowed me to see the effects of a change very quickly.

Experiment with Simple Examples

When I started developing Musicat, I used what I considered to be simple melodies, but much later I realized that I could make far simpler melodies. Running the program on extremely simple — indeed, nearly trivial — melodies often yielded surprising and key insights, while more complicated melodies often had a way of obscuring fundamental issues.

Have a Pressure-based Programming Mindset

Abhijit Mahabal coined the phrase “pressure-based programming” to describe his development of the Seqsee program. He and I spoke about this phrase only briefly, but I appreciate the idea. Programming a FARG model is quite unlike traditional programming because so much that happens is random. Instead of trying to think about the detailed sequence of events that might occur for desired structures to be built in the Workspace, it is helpful to think in terms of hundreds or thousands of codelets collectively exerting *pressure* on how structures will emerge over time in the Workspace.

During development, my use of a programming trick made it slightly easier for me to think in terms of pressures: I made it extremely fast and simple to add new codelets to the system using a technical feature of the C# language called a “class attribute”. A class attribute is a simple marker that can be attached to a class to provide useful information that the

program can use at runtime. I defined a “CodeletAttribute” that the Coderack could detect when it needed to populate itself with more codelets. Then, any time I wanted to create a new type of codelet, I simply wrote a new class (deriving from the base “Codelet” class), and attached the CodeletAttribute to the class, along with a default urgency level. Then, simply recompiling the library of codelet classes would make copies of this new codelet appear in the Coderack.

Keep Melody and Rhythm Separate

On several different occasions in the past several years, I have come across one particular way of simplifying musical problems: separating the two primary elements of music — melody and rhythm — from each other. RhythmCat, for example, totally ignored pitch and focused just on rhythm. In the Seek Well domain, on the other hand, all notes have the same duration but vary in pitch. Melody and rhythm can interact in complex ways, but temporarily treating them as independent components can be surprisingly effective.

ARCHITECTURAL CONTRIBUTIONS

Musicat is based on the FARG architecture, but it contributes several ideas that are novel with respect to other FARG programs. A short list of the most significant ideas follows.

Temporal Domain

Time is central to Musicat’s domain. Of course, in other FARG programs, time passes as the program runs, but Musicat is unique in that input is given to the program in real time. Seek Whence and Seqsee have a somewhat temporal nature because they process number sequences, and the numbers arrive one at a time, but there can be an arbitrarily long gap between one number and the next, so time is not nearly as critical a pressure. Musicat

works solely in the music domain, but some of its architecture and some aspects of its codelets might be applicable to other temporal domains.

Motivic Memory in a Slipnet

The motivic memory introduced in MusicatW can be seen as a rudimentary learning mechanism: when new motifs are heard, they are added to the Slipnet, which constitutes a long-term memory. (The Phaeaco program does something similar.) In MusicatW, new Slipnet nodes can be created by codelets, and these nodes can subsequently be used by the program even when running on different melodies. Motif notes in the Slipnet form a small network of relationships: if two motifs share a rhythmic pattern or a pitch pattern, then activation can spread from one motif to the other. Motivic memory, while not explored deeply in Musicat, may prove to be useful for managing constraints of working memory and real-time listening.

Dynamic Group Scoring

In Copycat, strength values are computed for each group using a weighted sum of scores determined by various features of the group. In Musicat, there are factors contributing to the strength of a group that are created dynamically based on context, so there is no finite list of features whose scores can be included in such a sum. Even if there were such a list, I find the idea that *all* features of a group could be included to be cognitively implausible. Rather, only a *subset* comprised of features noticed by the program (called Group Reasons if they contribute a positive value, and Group Penalties for negative values) is included in the sum. A “squashing function” is used to ensure that the strength cannot exceed the maximum value of 100. Thus, the strength of a group is determined not by a preordained and fixed set of group features, but rather by aspects of the group that have become salient to the program.

In my experience, scoring of perceptual structures is one of the most critical components of designing a FARG model, and it definitely deserves more attention in the design of future models.

Instant Feedback during Development

The “add-in” that I developed for my programming environment, described in Chapter 8, is not part of the FARG architecture *per se*, but I found it to be a very helpful tool. Programming FARG models is surprisingly different from typical software engineering, so new tools such as this one are essential for handling the unusual programming challenges that arise.

CONTRIBUTIONS TO THE MODELING OF MUSIC COGNITION

Musicat is based on many things: a collection of ideas from many different people in the field of music cognition; the FARG architecture and projects by my FARG predecessors; my own ideas and intuitions about music; and countless conversations with Douglas Hofstadter about music, analogies, and the design and the behavior of the program. The long process of designing and implementing the program and figuring out how to model music-listening in a FARG framework has resulted in numerous personal insights. Because my project builds on so much prior knowledge, and because it is still quite exploratory in nature, it may seem conceited or premature for me to include a section entitled “contributions to the modeling of music cognition”. My intent here is not to overstate my own contribution, but rather to pass along the most valuable ideas that have come up in the course of this work, in the hope that they will inspire further research. An analogy may be helpful: one of the activities I engaged in while designing Musicat was to create codelets that would constitute pressures for the system to hear music in a certain way that seemed reasonable to me;

similarly, the list of contributions in this section can be viewed as slight pressures for other researchers to focus on certain ways of modeling music cognition that seem important to me.

Listening as Performance

The most common question about Musicat I have heard from people who first hear about the program is simply “But what does it *do*? What is the output from the program?” The simplest answer — “It listens” — is typically unsatisfying to such people, but if I state that Musicat actively creates a “listening performance”, they generally seem surprised but interested. The idea that listening is an active process is, of course, not new (the idea that music relies on the three elements of composer, performer, and listener is well known), but the particular perspective of “listening as performance” might be unique in the field of music-cognition modeling. I consider “listening performances” to involve creative work and the active creation of highly complex mental structures by the listener.

Importance of Rhythm

Naturally, rhythm is important, but I was surprised at how much of an improvement I saw in Musicat after we modified it to focus its attention much more on rhythm than on pitch. Rhythm seems to be more primal in human perception of melodies, and it makes sense for a listening model to reflect this. Perhaps my surprise just reflects my own bias: I am personally more interested in melody and harmony than in rhythm. I encourage others who share this bias to be sure to give adequate attention to rhythm.

A Computer Implementation of “Perspects”

Adam Ockelford’s notion of musical perspectives seems to me to be quite amenable to inclusion in a computer model, especially one that creates analogies between musical

structures. Ockelford himself has done some music research that makes use of perspects in a quantitative manner (Ockelford, 2008), but in that work in they are restricted to the domain of rhythm, even though the perspect concept applies to many musical parameters. In MusicatW, perspects were important components of the model: codelets could perceive various aspects of notes and record these perceptions (perspects) in the Workspace. In addition, I generalized the notion of perspects to apply not only to individual notes but also to groups of notes.

Elaboration of the Intricacies of Listening to Simple Melodies

In my experience, research that models music-listening tends to focus on one of two different points along a spectrum of musical complexity. There is a body of work at the less-complex end of the spectrum that examines listening on a short time-scale that involves just a few notes, exemplified by Krumhansl and Kessler's experiments on pitch stability in a tonal context (Carol L. Krumhansl, 1990). At the other end of the spectrum, there is much research on listening in real-world (and larger-scale) contexts, such as work that studies listeners' emotional responses as they listen to an entire musical piece, such as a pop song or a classical work that may be several minutes long. Of course there are numerous exceptions, such as David Temperley's computer models (Temperley, 2001), but my impression is that there is relatively little attention paid to melodies of the sort that I included earlier in this dissertation in the "Simple Melodies" chapter. Melodies such as "Twinkle, Twinkle, Little Star" and "Sur le pont d'Avignon" are deceptively simple: a full description of a human listening performance for melodies of this sort would be an extremely deep contribution to the understanding of both music and the human mind. Listening to simple melodies is far subtler and more complex than one might suspect, and I hope that Musicat's listening

performances, both in their accomplishments and in their failures, give at least an inkling of this complexity.

The Importance of Analogy

In Chapters 2 and 4 I mentioned several instances of theorists discussing the importance of musical parallelism — recall that parallelism was characterized in GTTM as an important avenue for future work, and Narmour also stressed its importance in his Implication–Realization theory. To the best of my knowledge, no prior computer models of musical listening make use of parallelism except in an extremely simple manner. In this work, however, parallelism is central: Musicat’s analogies are a concrete implementation of the formerly only theoretical concept of parallelism. Musicat begins to fill the gap in theory explicitly mentioned by Lerdahl and Jackendoff, as I discussed in Chapter 4.

Moreover, I propose (admittedly without any justification aside from my own intuitions resulting from this work) that analogy is fundamental in music cognition, not just in that it helps to make sense of the structural organization of a piece of music, but also in that it contributes to musical semantics. To put it more clearly: grouping helps a listener understand music’s formal structure, and analogy is important in generating a grouping structure, but perceiving analogy also contributes to a listener’s sense of music having *meaning*.

Meta-grouping in Music-cognition Modeling

Computer models of music cognition often examine the problem of grouping. However, grouping is typically approached from the perspective of partitioning; in other words, group boundaries exist at only one level. Meta-groups or hierarchies are important to theorists, but appear less often than I would expect in computer models. For Musicat,

analogy is central, and all but the simplest analogies are formed between meta-groups, so meta-groups are essential. The set of Musicat's codelets that are involved with meta-group formation gives some ideas about how meta-groups might arise in listening, but much more work is needed. The field of music cognition would greatly benefit from an increased understanding of how listeners form meta-groups.

Musicat Introduces Structural Expectations

Melodic expectation has been a focus of much attention in the past decade or two. A great deal of work in the field involves expectation at the note level. That is, models typically generate expectations for the following note or several notes. In contrast, the latest version of Musicat generates expectations at the level of entire grouping structures and of inter-group analogies, rather than just at the level of individual notes. To be sure, both types of expectation are important, but these higher-level expectations translated more obviously into improvements of the program's listening performances.

Preliminary Steps towards a FARG-style Seek Well Program

In Chapter 2 I wrote about how the initial version of Musicat was based on Steve Larson's theory of musical forces and was intended as a successor to his Seek Well program. MusicatH attempted to automatically generate hierarchical embellishment structures, but the program didn't have much success. I believe that the main problem was the lack of analogy-making and of reliable group formation and meta-group formation in that program. Thus, although Musicat eventually diverged from the original goal of being an implementation of Larson's theory of musical forces using the FARG architecture, I now see some ideas in Musicat as necessary prerequisites to the automatic generation of embellishment structures. If

a future version of Musicat could successfully generate such structures, they could in turn be used to generate note-level expectations based on musical forces.

Future Work

Musicat is a work in progress, and there are many ways in which the program could be improved. The “Things Musicat Does Badly” section above suggests several obvious areas for additional work, as does careful observation of the program’s listening performances, as given in Chapters 5–7. In this final section, I prioritize a few items that I think would be important next steps for Musicat. Following that, I speculate briefly about longer-term implications of this work.

NEXT STEPS

My suggestions for future improvements are listed below in priority order; the first item has the highest priority, in my opinion.

1. Add more knowledge about pitch and tonality, especially focusing on types of tension associated with certain scale degrees, possibly in specific harmonic contexts.
2. Fix problems with the destruction and “resurrection” of large structures.
3. Remove the constraint that groups (and hence analogy components) must start and end at measure boundaries, while keeping a strong bias in favor of making group boundaries at these places.
4. Make the thicknesses of bar lines more responsive to strong groups that straddle strong bar lines. Grouping and bar-line thicknesses should influence each other more than they do at present.

5. Allow the program to “reset” its listening after an extremely thick bar line (*i.e.*, after reaching strong closure). Listening should be retrospective in some cases, but after a strong closure, listening seems to start “afresh”. This should help the program listen better to long melodies.
6. Get the program to notice certain key types of superficial features more readily and to stick to simple perceptions based on such features unless and until they are strongly contradicted.
7. Restore the motivic memory component of the program, which was removed after MusicatW.
8. Generate note-level expectations to complement the structural expectations currently generated. Examine the program’s real-time surprise and satisfaction based on how heard music denies or satisfies these expectations.

A HYPOTHESIS FOR FUTURE TESTING

After I have improved Musicat in the ways specified in the previous section, and in particular after removing the strict constraint that groups must start and end at measure boundaries (item 3 above), I hope to compare Musicat’s grouping performance with that of Temperley’s CBMS model. More importantly, I intend this comparison to substantiate the claim made in Chapter 4 that analogy-making is essential for modeling in music cognition. With the present version of the program, however, Musicat cannot be tested on the same corpus of melodies used by Temperley in his experiments (Temperley, 2001) because of the constraint on where group boundaries occur: in the corpus, many group boundaries occur at places other than measure boundaries. (See Appendix C, however, for some preliminary results obtained from Musicat if we consider a modified version of the problem.)

The CBMS model of grouping is intended to work on simple monophonic folk melodies, just as Musicat is, but uses just a few simple rules. An important difference is that grouping in CBMS is not hierarchical; instead it simply predicts phrase boundaries without regard to the hierarchical structure considered by GTTM or Musicat. In addition, CBMS has a notion of parallelism, but this should not be confused with the parallelism discussed in GTTM. In CBMS, “parallelism” is the very specific and simple idea that successive musical phrases tend to start at the same metric position, which is quite similar to Musicat’s strict restriction about group boundaries (when considered in conjunction with how Musicat internally shifts measures sideways to account for upbeats). CBMS prefers groupings which yield to this pressure of beginning successive phrases at the same metric position. However, the notion of parallelism invoked in GTTM, which operates naturally on hierarchical grouping structures instead of phrase boundaries, is the type of parallelism I am interested in with Musicat.

While much of the work of Musicat and other FARG models is that of modeling a cognitive system and examining whether the model has similar behaviors as humans, the difference in approach between Musicat and CBMS results in a straightforward, testable hypothesis involving analogy-making. CBMS is quite well-tuned to detecting phrase boundaries, and I hypothesize that if I disable Musicat’s analogy-making mechanisms, it will perform worse than CBMS with respect to detecting phrase boundaries for the corpus of melodies used in Temperley’s experiments. However, with analogy-making enabled, the predicted boundaries should be more accurate than those of CBMS.

Whither Musicat? (Questions and Speculations about the Future)

The ideas in this section look very far ahead and are thus very speculative, but nonetheless I hope they are thought-provoking. What can we look forward to from Musicat in the future?

COULD MUSICAT BE EXTENDED TO LISTEN TO POLYPHONIC MUSIC?

Polyphonic music is much more complex than monophonic music, yet I believe that the most significant problems occur in listening to a melody. Fundamental structures such as groups and analogies are also central to polyphonic listening, although grouping will need to be more flexible to allow for multiple melody lines and their interactions. Fortunately, Musicat is already based on the parallel actions of a large number of codelets, and this parallelism will be easily adaptable to the task of listening to multiple lines of music in parallel. As the degree of polyphony increases, a human listener's perceptual resources available for each individual note are limited; this sort of limitation will arise naturally in the model as well, because a fixed number of codelets is allocated to be run for each unit of musical time. Musicat will need better mechanisms for focusing its attention when there are many choices about what to listen to, but the current architecture seems adaptable to polyphonic listening.

WILL MUSICAT EVER LISTEN TO AUDIO RECORDINGS?

Musicat was designed to handle note-level input, not raw audio. However, since Musicat runs in real time, this offers hope for real-time audio listening. I can imagine a hybrid system in which Musicat "listens" to notes that are detected by a lower-level audio processing system based on standard engineering techniques such as the fast Fourier

transform, filter banks, Cepstral coefficients, and the like. However, cutting-edge techniques available at the time of this writing still make many errors in detecting notes. I believe that a listening program such as Musicat could provide real-time feedback to the audio-processing component of such a system, potentially improving the accuracy of note detection. Just as Musicat's own top-down pressures can guide its low-level perception, so Musicat as a whole might act as a source of top-down pressures that could guide the extremely low-level perception provided by an audio-processing algorithm. (After all, the human auditory system has neural feedback loops from higher-level processing back down to lower-level neural circuits, suggesting that higher-level cognition has a very direct way to focus low-level auditory perception.)

WILL MUSICAT EVER HEAR MUSIC IN THE SAME WAY AS A HUMAN DOES?

As much as I would like to imagine Musicat developing into a full-fledged listener, the honest answer is “no”! I have high hopes for Musicat, but one critical thing is missing: Musicat doesn't have emotions. I haven't discussed emotion in this dissertation because it is simply beyond the scope of this work, but emotion is critical to the human listening experience. Human listeners can be moved to tears or experience “chills” (or “*frisson*”) as a result of listening to music. Until someone can model emotion, Musicat will remain unmoved by music, no matter how well it understands its structure and makes analogies.

WILL MUSICAT EVER BECOME A COMPOSER?

I am slightly more hopeful here (or perhaps I'm just a cockeyed optimist). As long as it doesn't listen in the same emotional way as a human does, Musicat will always lack something important. However, there are some interesting results in computer music composition, especially David Cope's program EMI (Cope & Hofstadter, 2001), which,

despite its shortcomings, succeeds in making stuff that sounds to many people like meaningful music. Programs such as EMI, however, lack the ability to listen to what they compose! If Musicat is one day able to create passable listening performances, it would have a huge advantage over any “deaf” computer composers, and it might well try its “hand” (and “ears”) at writing music!

APPENDIX A

References

- Bernstein, L. (1976). *The Unanswered Question: Six Talks at Harvard*. Cambridge, Mass.: Harvard University Press.
- Butler, D., & Brown, H. (1994). Describing the Mental Representation of Tonality in Music. In R. Aiello & J. Sloboda (Eds.), *Musical Perceptions*. New York: Oxford University Press.
- Byrd, D. (1984). *Music Notation by Computer*. (Ph.D. Dissertation, Computer Science), Indiana University.
- Cadwallader, A.C., & Gagné, D. (1998). *Analysis of Tonal Music: A Schenkerian Approach*. New York: Oxford University Press.
- Cambouropoulos, E. (1997). Musical Rhythm: A Formal Model for Determining Local Boundaries, Accents, and Metre in a Melodic Surface. In M. Leman (Ed.), *Music, Gestalt, and Computing: Studies in Cognitive and Systematic Musicology* (pp. 277–291). Berlin: Springer-Verlag.
- Carlsen, J.C. (1981). Some Factors Which Influence Melodic Expectancy. *Psychomusicology*, 1, 12–29.
- Cooper, G., & Meyer, L.B. (1960). *The Rhythmic Structure of Music*. Chicago: University of Chicago Press.
- Cope, D. (1996). *Experiments in Musical Intelligence*. Madison, Wisc.: A-R Editions.
- Cope, D., & Hofstadter, D.R. (2001). *Virtual Music: Computer Synthesis of Musical Style*. Cambridge, Mass.: MIT Press.
- Copland, A. (1960). *Copland on Music* (1st ed.). Garden City, N.Y.: Doubleday.
- Deutsch, D., & Feroe, J. (1981). The Internal Representation of Pitch Sequences in Tonal Music. *Psychological Review*, 88(6), 503–522.
- French, R.M. (1992). *Tabletop: An Emergent, Stochastic Computer Model of Analogy- Making*. The University of Michigan.
- Glover, F., & Laguna, M. (1997). *Tabu Search*. Boston: Kluwer Academic Publishers.

- Hirata, K., Hamanaka, M., & Tojo, S. (2007). *Techniques for Implementing the Generative Theory of Tonal Music*. Paper presented at the International Conference on Music Information Retrieval, Vienna, Austria.
- Hofstadter, D.R. (2001). Analogy as the Core of Cognition. In D. Gentner, K. J. Holyoak & B. N. Kokinov (Eds.), *The Analogical Mind: Perspectives from Cognitive Science* (pp. 499–538). Cambridge, MA: MIT Press.
- Hofstadter, D.R., & Fluid Analogies Research Group. (1995). *Fluid Concepts & Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*. New York: Basic Books.
- Hofstadter, D.R., & Sanders, E. (2013). *Surfaces & Essences*: Basic Books.
- Hugo, V. (1862). *Les Misérables*. Paris, France: Pagnerre.
- Hugo, V. (1987). *Les Misérables* (L. M. Fahnestock, Norman, Trans.). New York, New York: Signet Classic.
- Huron, D.B. (2006). *Sweet Anticipation: Music and the Psychology of Expectation*. Cambridge, Mass.: MIT Press.
- Jones, M.R. (1993). Dynamics of Musical Patterns: How do Melody and Rhythm Fit Together? In T. J. Tighe & W. J. Dowling (Eds.), *Psychology of Music: The Understanding of Melody and Rhythm* (pp. 67–92): Lawrence Erlbaum Associates.
- Kanerva, P. (1988). *Sparse Distributed Memory*. Cambridge, Mass.: MIT Press.
- Kasimi, A., Nichols, E., & Raphael, C. (2007). *A Simple Algorithm for Automatic Generation of Polyphonic Piano Fingerings*. Paper presented at the International Conference on Music Information Retrieval (ISMIR), Vienna, Austria.
- Krumhansl, C.L. (1990). *Cognitive Foundations of Musical Pitch*. New York: Oxford University Press.
- Krumhansl, C.L., & Kessler, E.J. (1982). Tracing the Dynamic Changes in Perceived Tonal Organization in a Spatial Representation of Musical Keys. *Psychological Review*, 89(4), 334–368.
- Larson, S. (1993a). Computer Models of Melodic Implication and Key Determination in Tonal Music. Society for Music Perception and Cognition, Philadelphia, Available as CRCC Technical Report #77.
- Larson, S. (1993b). Seek Well: A Creative Microdomain for Studying Expressive Meaning in Music. Center for Research on Concepts and Cognition; 510 North Fess; Bloomington, IN 47408.
- Larson, S. (1997a). Continuations as Completions: Studying Melodic Expectation in the Creative Microdomain *Seek Well*. In M. Leman (Ed.), *Music, Gestalt, and Computing: Studies in Cognitive and Systematic Musicology* (pp. 321–334). Berlin: Springer-Verlag.
- Larson, S. (1997b). The Problem of Prolongation in *Tonal Music*: Terminology, Perception, and Expressive Meaning. *Journal of Music Theory*, 41(1), 101–136.
- Larson, S. (2004). Musical Forces and Melodic Expectations: Comparing Computer Models and Experimental Results. *Music Perception*, 21(4), 457–498.

- Larson, S. (2012). *Musical Forces: Motion, Metaphor, and Meaning in Music*. Bloomington: Indiana University Press.
- Lartillot, O. (2002). Generalized Musical Pattern Discovery by Analogy from Local Viewpoints. *Discovery Science, Proceedings*, 2534, 382–389.
- Lartillot, O. (2004). A Musical Pattern Discovery System Founded on a Modeling of Listening Strategies. *Computer Music Journal*, 28(3), 53–67.
- Lartillot, O. (2005). Multi-dimensional Motivic Pattern Extraction Founded on Adaptive Redundancy Filtering. *Journal of New Music Research*, 34(4), 375–393.
- Lerdahl, F. (2001). *Tonal Pitch Space*. New York: Oxford University Press.
- Lerdahl, F., & Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. Cambridge, Mass.: MIT Press.
- Linhares, A. (2005). An Active Symbols Theory of Chess Intuition. *Minds and Machines*, 15(2), 131–181.
- Linhares, A. (2008). The Emergence of Choice: Decision-Making and Strategic Thinking through Analogies: EBAPE / Getuilo Vargas Foundation.
- Linhares, A., & Freitas, A.E.T.A. (2010). Questioning Chase and Simon's (1973) "Perception in Chess": The "Experience Recognition" Hypothesis. *New Ideas in Psychology*, 28(1), 64–78.
- Margulis, E.H. (2005). A Model of Melodic Expectation. *Music Perception*, 22(4), 663–714.
- Meyer, L.B. (1956). *Emotion and Meaning in Music*. Chicago: University of Chicago Press.
- Miller, G. (1956). The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information. *Psychological Review* 63 (2), 81–97.
- Miller, G.A. (1995). Wordnet — a Lexical Database for English. *Communications of the ACM*, 38(11), 39–41.
- Mitchell, M. (1993). *Analogy-making as Perception: A Computer Model*. Cambridge, Mass.: MIT Press.
- Narmour, E. (1990). *The analysis and cognition of basic melodic structures : the implication-realization model*. Chicago: University of Chicago Press.
- Narmour, E. (1992). *The analysis and cognition of melodic complexity : the implication-realization model*. Chicago u.a.: Univ. of Chicago Pr.
- Nichols, E., & Raphael, C. (2006). *Globally Optimal Audio Partitioning*. Paper presented at the International Society for Music Information Retrieval, Victoria, Canada.
- Ockelford, A. (1991). The Role of Repetition in Perceived Musical Structures. In P. Howell, R. West & I. Cross (Eds.), *Representing Musical Structure* (pp. 129–160): Academic Press.
- Ockelford, A. (2004). *Repetition in Music: Theoretical and Metatheoretical Perspectives*. Aldershot, Hants, England, Burlington, VT: Ashgate.
- Ockelford, A. (2008). *In the Key of Genius: The Extraordinary Life of Derek Paravicini*. London: Arrow.

- Povel, D.-J., & Essens, P. (1985). Perception of Temporal Patterns. *Music Perception*, 2, 411–440.
- Raphael, C., & Nichols, E. (2009). Linear Dynamic Programming and the Training of Sequence Estimators. In J. Chinneck, B. Kristjansson & M. Saltzman (Eds.), *Operations Research and Cyber-Infrastructure* (pp. 219–231). New York, NY: Springer.
- Rothstein, W.N. (1989). *Phrase Rhythm in Tonal Music*. New York: Schirmer Books; Collier Macmillan Publishers.
- Schellenberg, E.G. (1997). Simplifying the Implication–Realization Model of Melodic Expectancy. *Music Perception*, 14, 295–318.
- Schoenberg, A. (1967). *Fundamentals of musical composition* (G. S. Strang, Leonard Ed.). New York,: St. Martin's Press.
- Snyder, B. (2000). *Music and Memory: An Introduction*. Cambridge, Mass.: MIT Press.
- Temperley, D. (2001). *The Cognition of Basic Musical Structures*. Cambridge, Mass.: MIT Press.
- Westergaard, P. (1975). *An Introduction to Tonal Theory*: W. W. Norton & Co.
- Yeston, M. (1976). *The Stratification of Musical Rhythm*. New Haven: Yale University Press.

APPENDIX B

Pilot Study

In this appendix I briefly describe the pilot study mentioned in Chapter 9. I found the results interesting, although I didn't end up continuing this line of research in Musicat. With Steve Larson's help I designed an experiment in which 14 subjects — mostly undergraduate or graduate students from the Jacobs School of Music at Indiana University, with the others having significant musical training — were asked to listen to two notes and to sing an improvised melody beginning with those two notes. There were 25 possible note pairs used as cues in the experiment: all 12 possible intervals, from the unison to the octave, were possible, and furthermore, each interval (with exception of the unison) could be presented in either the ascending or descending direction. For example, a participant might hear the two notes C–F (ascending) or D–C (descending) and then sing a melody that started with those two notes. For each participant, all 25 cues were transposed to lie in a comfortable vocal range (based on self-identification of bass, tenor, alto, or soprano voice type), and these transposed cues were presented in random order. Each participant performed this entire procedure two times, resulting in 50 improvised responses per participant. My research assistant Elton Joe transcribed the sung responses into standard music notation, and included the key and meter that were implied by the performance.

Participants were asked to sing the same improvised melody twice in a row, which helped solidify the key and meter, facilitating the transcription task. Also see (Carlsen, 1981) for a very similar but larger-scale experiment with 91 participants and 15 repetitions of the task for each person; Carlsen's data analysis had a different focus than mine (he looked only at pitch expectation and tonal function, not at meter); otherwise, his results would have been sufficient.

My hypothesis was that the choice of initial interval would influence both the key (with respect to the first note) and the meter of the resulting improvised melody. For instance, I expected that given an ascending step as the cue, such as C–D, most people would hear the C as the tonic, and generate a response in C major. Furthermore, I expected people to sing responses in 4/4 meter the majority of the time for this cue. However, given an initial ascending perfect fourth such as D–G as the cue, I expected people to hear the G as the tonic, and also to hear the G as a downbeat, with the D as a pickup note.

I examined several features in the data resulting from the experiment:

- Effect of the initial interval on the inferred key;
- Effect of the initial interval on the inferred meter and metric placement of the interval;
- Percentage of time that the first note is heard as the tonic; and
- Tonal distribution of the first several response notes.

Indeed, there were interesting effects (but because this was only a pilot study with a very small number of participants, I make no claims about its significance). As in Carlsen's study, the initial intervals had both tonal and rhythmic implications.

Meter is usually based on 2 or 4 beats per measure (duple meter) or 3 beats per measure (triple meter). (For simplicity in this analysis, I also treat compound meters such as

6/8 as members of the category “triple meter”.) In the experiment, 79% of the responses were in duple meter; 21% were in triple meter. This is similar to the results of other studies that show a default bias for Western tonal-music listeners to expect duple meter. However, for the initial interval of an ascending half step (such as B–C, ascending), 32% of the responses were in triple meter — an 11% increase. Similarly, the ascending perfect fourth and descending perfect fifth each resulted in triple meter for 26% of the responses — a 5% increase over the baseline. For all three of these cases, the second note of the interval is often perceived as the tonic. For example in the ascending B–C interval, the B is often heard as the leading tone to a tonic C.

Why did these intervals increase the tendency for people to generate responses in triple meter? The data suggests a simple explanation. In these cases where the second note was heard as the tonic, it was more likely than normal to be heard as a downbeat (and hence the first note heard as an upbeat). Specifically, consider the following select group of seven interval types that lead to an increased proportion of responses with an upbeat:

Descending: perfect fifth

Ascending: half-step, perfect fourth, tritone, minor sixth, major sixth, minor seventh

In this group of intervals, the first note was heard as an upbeat 22% of the time, in contrast to the other 18 interval types, which were heard as starting with an upbeat only 10% of the time. In terms of tonal function, the first note was heard as something other than the tonic a whopping 84% of the time, in contrast to 57% for intervals not in this group. Therefore, it seems clear that for these intervals in particular, the first note doesn’t usually sound like a tonic. Often, then, the second note is heard as the tonic. This leads the listener to hear the first note as an upbeat more of the time than normal. Finally —here’s the point — hearing the first note as an upbeat is positively correlated with hearing the passage in triple meter

(given an upbeat hearing of the first note, triple meter responses occur 30% of the time, versus 19% if the first note was heard as a downbeat). This is not too surprising, because upbeats are a common feature of triple-meter melodies. For intervals in our select group, triple meter occurs 29% of the time, versus 17% for non-group intervals.

Looking at the data in general, I find that if an interval is heard both as starting with an upbeat and as starting with a non-tonic note, the response is in triple meter 33% of the time (in contrast to the 21% baseline). The data strongly suggest relationships between the initial interval, inferred tonal function, and inferred meter. This is important because it illustrates how tonal function and meter are inexorably linked. A future version of Musicat that is not given the key and meter *a priori*, but must infer them on its own, will need to account for these complexities.

Rhythmic considerations aside, I also looked at some simple frequency distributions of the tonal function of the first notes of the responses. The first note was heard as the tonic 35% of the time. The second note was heard as the tonic 30% of the time. The third note (chosen by the participant) was the tonic 15% of the time. The following diagram shows the frequency of each tonal function for the first note of each response.

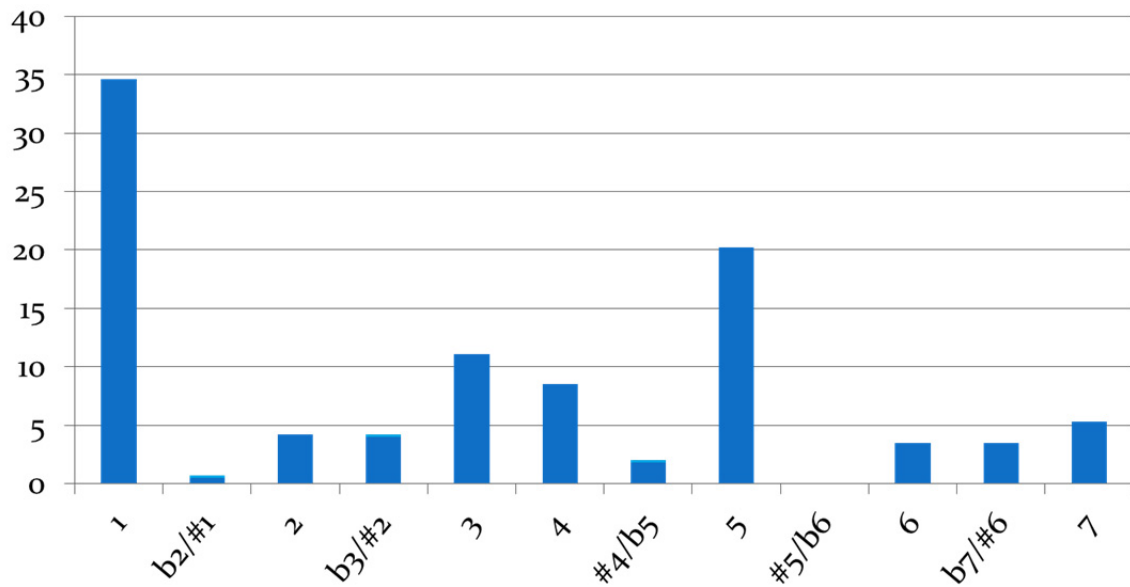


Figure 10.1: Frequency counts of tonal functions for first notes.

The picture in Figure 10.1 is evocative of the Krumhansl and Kessler (1982) key profiles, reproduced in the following figure for both major and minor modes:

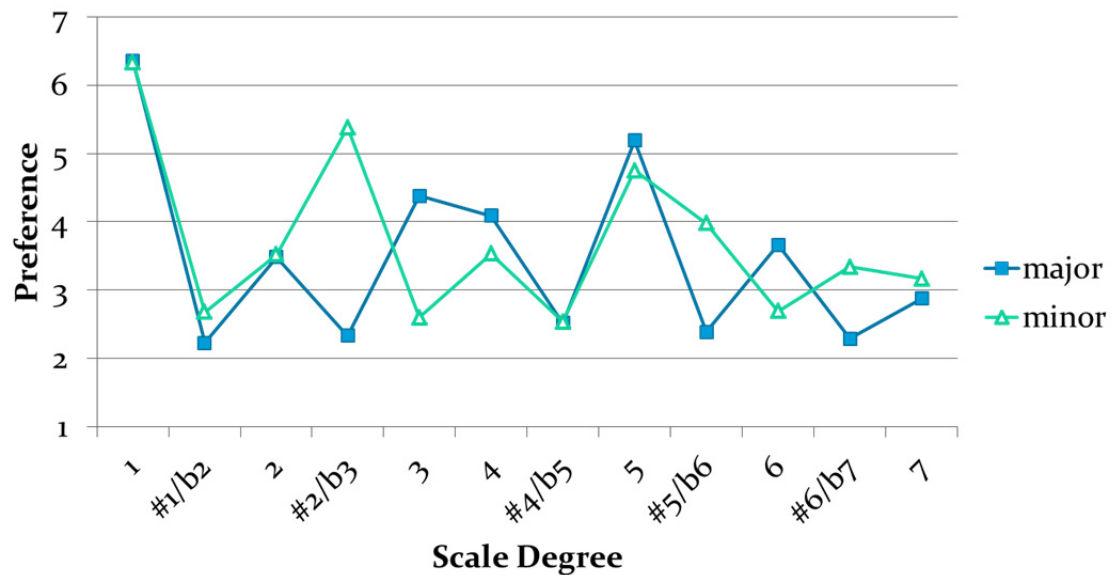


Figure 10.2: Krumhansl and Kessler's key profiles.

I compared my results based on frequency of scale degree in the first notes of responses to the key profiles in Figure 10.2 by generating a single key profile from the Krumhansl and Kessler (1982) data, weighting the major profile by 0.72 and the minor profile by 0.28, corresponding to the rate of major and minor mode responses in my data. Then I scaled the frequencies from Figure 10.1 to match the 1–7 “preference” range. The resulting graph follows:

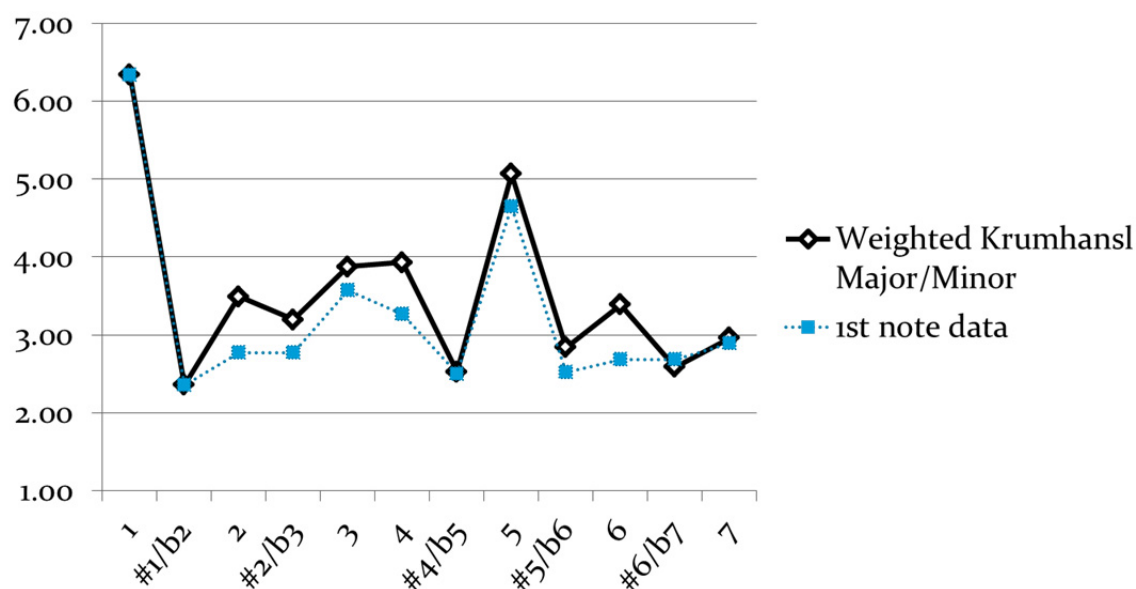


Figure 10.3: Comparison of first note data with weighted Krumhansl and Kessler profiles.

The shapes of the two lines in Figure 10.3 are roughly the same. This is interesting, because whereas the Krumhansl and Kessler profiles are based on preference ratings for scale degrees in a tonal context, my data are based solely on frequency counts of inferred scale degrees for the first note of an improvised melody.

I made a similar graph including results from the second and third notes, respectively (Figure 10.4).

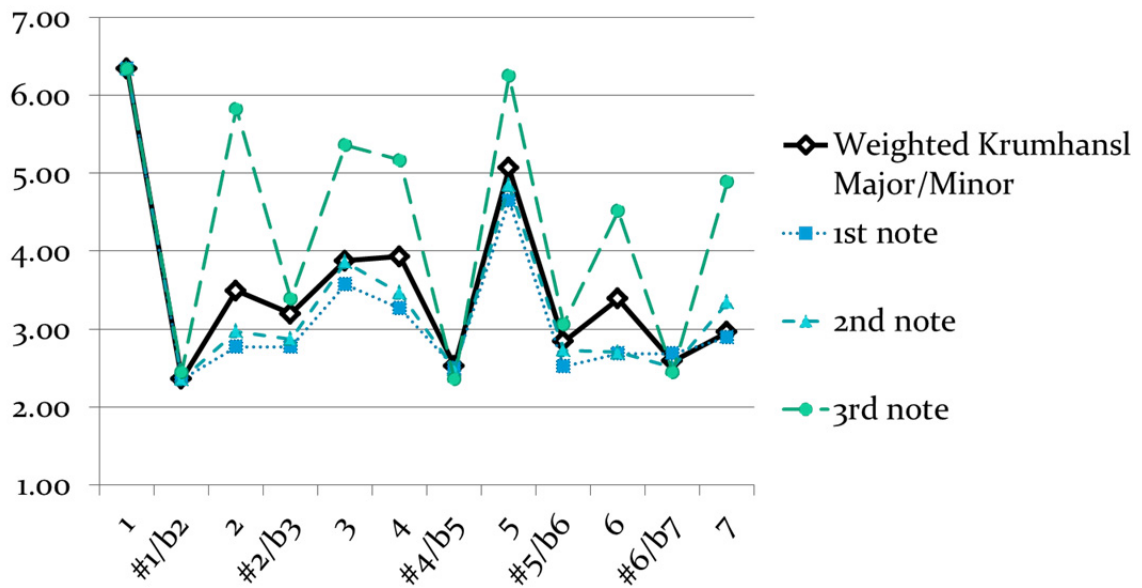


Figure 10.4: More comparisons.

APPENDIX C

Preliminary Quantitative Results

In Chapter 10, I proposed that after additional improvements to Musicat, it would be feasible to compare it with Temperley's model from *The Cognition of Basic Musical Structures* (CBMS) (Temperley, 2001). Because the current version of Musicat is required to put group boundaries at measure boundaries (taking into account Musicat's shifting of entire melodies to account for pickup bars), it is impossible to compare the grouping structures generated by Musicat and the CBMS grouping program in a way that puts the two programs on equal footing. The Essen folksong corpus, which was used in Temperley's reported results from CBMS, includes many instances of group boundaries that are not found at measure boundaries (even if we account for pickup bars). Temperley's program does not have the measure-boundary restriction on grouping that Musicat does.

Because of this restriction in Musicat, I generated a simplified version of Temperley's 65-song subset of the Essen corpus, in which all mid-measure group boundaries were "rounded" to occur at the start of the measure in which they occurred (after shifting all barlines sideways, if necessary, to account for pickup bars). For many melodies this has little effect; for others it makes Musicat's grouping boundaries more likely to line up with the ground truth. This simplification, to be sure, will artificially improve Musicat's reported

results to some degree, but I decided to include these preliminary results in any case, as a sneak preview of future testing to be done.

One other change was made to the ground truth file: several of the 65 melodies used by Temperley in testing CBMS were uninterpretable by Musicat, because they included notes of durations that Musicat could not handle. These melodies were excluded from this test; Musicat was tested on a subset of Temperley's subset of the Essen corpus.

I stress that these changes mean that the following comparison artificially inflates Musicat's results relative to those of CBMS, but perhaps not to a very large degree. Future testing, on even footing, is imperative, and will be carried out once I have improved Musicat and removed the group-boundary restriction.

The following table gives results obtained by running Musicat 20 times on each melody in the dataset, with a different random seed for each run. The boundaries indicated in the Essen corpus were converted into groups in a straightforward manner in order to generate a ground truth file, but no meta-groups were available, so all meta-groups generated by Musicat were ignored. I also included results for Musicat running on the Simple Melodies and Complex Melodies from chapters 6–7 of this thesis. Many of these melodies were used during the development of Musicat; this table shows results from testing on the training data for the Simple and Complex melody sets, and hence these values are also higher than they would be on a completely separate test set (Musicat was never run on the Essen subset during development, fortunately). For the Simple and Complex melodies, I have also included desired meta-groups and desired analogies in the ground truth.

I also report the number of groups and analogies that are “extra” for each run. The percentages in the table for extra groups or analogies were calculated by dividing the number of groups or analogies that were generated during a run but that did not exist in the ground truth by the total number of groups or analogies generated by the program for the same run.

	Groups correct	Groups extra	Analogies correct	Analogies extra
Musicat: Simple Melodies	83%	14%	48%	66%
Musicat: Complex Melodies	68%	43%	27%	78%
Musicat: Simplified Essen sub-subset	74%	39%	n/a	n/a
CBMS: Essen subset	76%	25%	n/a	n/a

Table 3: Preliminary results for Musicat, compared with reported results from CBMS.

Musicat does better on the simple melodies than on the complex melodies for all four metrics that were computed. Notice that many extra groups and analogies were generated, especially for the complex melodies. However, the scoring function I used treats groups and analogies in a binary fashion: a very weak group is weighted just as highly as a very strong group, so if very many extraneous weak structures are generated, they will have a large negative effect on the performance, even though those structures may not be very important to the program. A more fair test might use a threshold to remove weak structures from the calculation (just as is implemented in the user interface by the detail slider).

On the simplified subset of the Essen subset used by Temperley, Musicat generated 74% of the desired groups, trailing behind CBMS just slightly in performance. However, recall that these numbers are artificially inflated for Musicat. Even so, the numbers are encouraging to me, because it seems that Musicat is in striking distance of the accuracy of CMBS. Musicat and CMBS are both cognitively inspired models, but whereas CBMS is an off-line algorithm, Musicat generates groups in real-time, which is arguably more difficult. I was encouraged to see Musicat's 74% number, although a more fair comparison must be made in the future.

Curriculum Vitae

Eric P. Nichols
epnichols@gmail.com

Education

Indiana University — Bloomington

Joint Ph.D. in Computer Science and Cognitive Science,
Certificate in Cognitive Modeling, Ph.D. Minor in Music
Thesis advisor: Dr. Douglas R. Hofstadter

May 2006 – December 2012

M.S., Computer Science

January 2004 – April 2006

Montana State University — Bozeman

September 1995 – May 2000

B.S., Mathematics and Applied Mathematics with High Honors,
Minor in Computer Science

Institut National des Sciences Appliquées — Lyon, France

January 1997 – May 1997

Undergraduate Exchange in Computer Science and Mathematics

Research Experience

Center for Research on Concepts and Cognition (CRCC)

January 2004 – present

Developed a computer model of music perception and cognition in Western tonal music. Organized monthly meetings of the Fluid Analogies Research Group (FARG) at CRCC.

National Science Foundation CreativeIT Grant (IIS-0738384)

September 2007 – February 2010

Wrote a successful NSF grant proposal to fund my research in modeling musical expectation. The grant paid for my tuition and travel and also allowed me to purchase more equipment and hire and train part-time help to run experiments in music cognition.

Music Informatics | Indiana University

September 2005 – present

Designed and implemented novel algorithms for several music informatics problems with Dr. Christopher Raphael, including:

- Cost function optimization in a trellis graph (with applications in automatic harmonic analysis and piano fingering) (using C# and Java)
- Polyphonic audio transcription (using C)
- Audio segmentation (using C)
- Automatic melody and rhythm generation (using Python and C#)

Industry Experience

Software Engineering Intern | Google Research

June 2011 – August 2011

Improved automatic ranking of YouTube Slam music videos by 0.5% by developing new music-specific features for audio analysis—these features estimate singing quality, including how well a person is singing in-tune. Added the intonation feature to the upcoming Golden Set 5.0 release, so it will be computed for all uploaded YouTube videos. Developed a classifier to estimate singer age and demonstrated a heuristic for identifying talented young artists. Wrote a paper on this work intended for presentation at the ISMIR 2012 conference.

Research Intern | Microsoft Research, Theory and Computational
User Experiences (CUE)

June 2008 – August 2008

Solved difficult problems in automatic genre-specific harmonization of sung melodies, using probabilistic graphical models and a novel clustering method. Brought together researchers from the Theory, CUE, and Natural Languages groups to overcome modeling obstacles and prototyped new features for the Songsmith project inspired by these discussions. Ran user studies to test our new algorithms and interface, and presented the results at a conference on Intelligent User Interfaces.

President/Owner | AiMusic, LLC

January 2007 – present

Created and self-funded a startup specializing in creativity support tools for musicians. Developed multiple unique and profitable iPhone apps, including the first rhyming dictionary for iPhone, “Perfect Rhyme” (<http://tinyurl.com/perfectrhyme>).

Consultant | Retronyms, LLC

September 2007 – present

Designed and coded the server software for the award-winning GPS-based iPhone game “Seek ‘n Spell” (www.seeknspell.com). Developed a Windows Media Player plug-in (C++) to apply virtual surround-sound effects to an audio stream. Helped design a novel social website using Django.

Software Engineer | IntelliChem Inc.

May 2001 – December 2003

Second engineer hired by the founders in a successful, fast paced startup designing software for top-ten pharmaceutical companies; IntelliChem was acquired shortly after our successful 4.0 release. The company grew to comprise over 40 people (fifteen software engineers) by the time I left to start graduate school.

- Designed and implemented client applications (VB 6), web services (C#), and control software for robotic liquid handlers (C#).
- Managed and mentored offshore software engineers and summer interns working on my team to develop machine learning algorithms for chemical property prediction; I reported directly to our CEO for this mission-critical project.
- Developed custom software extensions, working directly with clients at Bristol-Myers Squibb, Merck, and Pfizer, owning the project all the way from the detailed requirements-gathering phase through testing and final delivery.

Lead Technical Writer | Cylant Technology

August 2000 – April 2001

Wrote both technical and user documentation for prototype software engineering and security tools in a demanding startup environment. Managed information-gathering sessions and served as the liaison between the technical team and the documentation and public relations teams. Provided mathematical and algorithmic guidance to the development team. Designed the company website.

Awards

- First Place in the Indiana University School of Informatics poster competition, for "Lyric-Based Rhythm Suggestion", 2009
- Seek 'n Spell game for iPhone: Best Use of Technology award, Come Out & Play Festival, New York City, 2009
- NSF SGER Grant for: "Musicat: a computational model of creativity in a musical domain," under the direction of Douglas R. Hofstadter, 2007-2009
- Summer Research Fellowship in Cognitive Science, Indiana University, 2004-2006
- NSF Graduate Research Fellowship Honorable Mention, 2003-2004
- Barry Goldwater Fellowship, 1998-1999
- Presidential Scholar, Montana State University, 1995-2000

Teaching at Indiana University

- Instructor for the graduate seminar Introduction to Music Informatics January 2011 – May 2011
- Instructor for an undergraduate artificial intelligence course September 2008 – December 2008
One of my former students recently asked my permission to make Othello tournament engine I developed for the course was recently refactored and open-sourced by a former student:
<http://tinyurl.com/GitHubOthello>

Skills / Other

- C# and .NET, Python, C, C++, Objective-C and iOS, Java, Django, Visual Basic 6, Scheme, Lisp, PERL, Ada, Pascal, and Assembly
- Keyboardist, bass guitarist, vocalist, and songwriter with the folk-rock band Angerhard Eclectic
- Composer of several modern classical music pieces
- Studied artificial intelligence and mathematics in Lyon, France for one semester; competent in the French language
- Studied the Portuguese language; derived some interesting heuristics about Portuguese noun gender using machine learning
- Selected by my advisor, Doug Hofstadter, to join him on his sabbatical in Paris (January – June 2010) while working on my research
- World traveler with a network of international friends – spent time in many different countries: France, The Netherlands, Sweden, Finland, Japan, Morocco, England, Canada, Germany, Italy, Switzerland, and Austria.
- Active dancer (salsa, swing, and contradance), cyclist, runner, racquetball player, and yoga student

Publications

Hughes, J., Lohman, B., Deckert, G., Nichols, E., Settles, M., Abdo, Z., and Top, E., "The role of clonal interference in the evolutionary dynamics of plasmid-host adaptation," *mBio* 3(4):e00077-12. doi:10.1128/mBio.00077-12. 2012.

Raphael, C., and Nichols, E., "Linear Dynamic Programming and the Training of Sequence Estimators," in *Operations Research and Cyber-Infrastructure*, eds. Chinneck, J., Kristjansson, B., and Saltzman, M. Springer: New York, NY, 2009, pp. 219–231.

Nagoski, E., Janssen, E., Lohrmann, D., and Nichols, E. "Risk, Individual Differences, and Environment: An Agent-Based Modeling Approach to Sexual Risk-Taking." Accepted for publication in Archives of Sexual Behavior.

Nichols, E., "Dynamic Melodic Expectation in the Creative Microdomain SeekWell." 2007. Available (from Center for Research on Concepts and Cognition; 510 North Fess; Bloomington IN 47408) as CRCC Technical Report #138.

Jordan, R., Nichols, E., and Cunningham, A. (1999). "The role of (bio)surfactant sorption in promoting the bioavailability of nutrients localized at the solid-water interface." *Wat. Sci. Technol.* 39(7), p. 91.

Conference Papers/Presentations

Nichols, E., DuHadway, C., Aradhye, H., and Lyon, R. "Automatically Discovering Talented Musicians with Acoustic Analysis of YouTube Videos," *IEEE International Conference on Data Mining (ICDM)*. December 2012, Brussels, Belgium.

Knopke, I. and Nichols, E. "Melodic Search and Pattern Discovery for Symbolic Music Information Retrieval", Tutorial at the International Society for Music Information Retrieval Conference (ISMIR). August 2010, Utrecht, The Netherlands.

Nichols, E. "Cantonese Melody-Composition Assistant," IPSJ Special Interest Group on MUSic and computer (SIGMUS), November 2009, Tokyo, Japan.

Nichols, E., Morris, D., Basu, S., and Raphael, C. "Relationships Between Lyrics and Melody in Popular Music," in International Society for Music Information Retrieval Conference (ISMIR), pp. 471–476, October 2009, Kobe, Japan.

Nichols, E. and Byrd, D. "The Future of Music IR: How Do You Know When a Problem is Solved?" in International Society for Music Information Retrieval Conference (ISMIR), October 2009, Kobe, Japan.

Nichols, E. "Lyric-Based Rhythm Suggestion," in International Computer Music Conference (ICMC), August 2009, Montreal, Canada.

Nichols, E. and Joe, E. "Modeling Meter and Key Implication," in Society of Music Perception and Cognition Conference (SMPC), August 2009, Indianapolis, Indiana, USA.

Nichols, E. "Lyric-Based Rhythm Suggestion," School of Informatics Grad Poster Session, April 17, 2009, Bloomington, Indiana, USA.

Nichols, E. "Musicat: a Computer Model of Music Cognition," New Voices in Academia Conference, March 27, 2009, Bloomington, Indiana, USA.

Nichols, E., Morris, D., and Basu, S. "Data-Driven Exploration of Musical Chord Sequences," in 13th International Conference on Intelligent User Interfaces (IUI), February 2009, pp. 227–236, Sanibel Island, Florida, USA.

Nichols, E. and Hofstadter, D.R. "Musicat: A Computational Model of Creativity in a Musical Domain," NSF CreativeIT Workshop, January 2009, Arlington, Virginia, USA.

Raphael, C., and Nichols, E., "Linear Dynamic Programming and the Training of Sequence Estimators," INFORMS Computing Society (ICS), 2009, Charleston, South Carolina, USA.

Linhares, A. and Nichols, E., "Automated Scientific Discovery and Hofstadter's Fluid Concepts Model," AAAI 2008 Fall Symposium on Automated Scientific Discovery, 2008, Arlington, Virginia, USA.

Nichols, E. and Linhares, A., "Creativity veRsus Classical Computation," North American Conference on Computing and Philosophy, 2008, Bloomington, USA.

Raphael, C. and Nichols, E., "Training Music Sequence Recognizers with Linear Dynamic Programming," in MML 2008 International Workshop on Machine Learning and Music, 2008, pp. 19–20, Helsinki, Finland.

Knopke, I. and Nichols, E., "Constrained Automatic Chord Alignment and Detection of Musical Structure," CIM 2008 Fourth Conf. on Interdisciplinary Musicology Proc., 2008, Thessaloniki, Greece.

Nichols, E. and Raphael, C., "Automatic Transcription of Music Audio Through Continuous Parameter Tracking," in ISMIR 2007 Eighth Int. Conf. on Music Inf. Retr. Proc., 2007, pp. 387–392, Vienna, Austria.

Kasimi, A., Nichols, E., and Raphael, C., "A Simple Algorithm for Automatic Generation of Polyphonic Piano Fingerings," in ISMIR 2007 Eighth Int. Conf. on Music Inf. Retr. Proc., 2007, pp. 355–356, Vienna, Austria.

Nichols, E. and Knopke, I., "Musical Attractors: A New Method for Audio Synthesis," in AES 31st Int. Conf. Proc., June 25–27, 2007, London, UK.

Nichols, E. and Raphael, C., "Globally Optimal Audio Partitioning," in ISMIR 2006 Seventh Int. Conf. on Music Inf. Retr. Proc., 2006, pp. 202–205, Victoria, Canada.