UNIVERSITY OF CALIFORNIA
Santa Barbara

Structural Metrics: An Epistemology

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Media Arts and Technology

by

Michael Benjamin Winter

Committee in charge:
Professor Clarence Barlow, Chair
Professor Curtis Roads
Professor JoAnn Kuchera-Morin
Professor Azer Akhmedov

June 2010

The dissertation of Michael Benjamin Winter is approved:

Chair Date

Date

Date

Date

University of California, Santa Barbara

June 2010

Structural Metrics: An Epistemology

To my father,

Eugene Julian Winter,

Acknowledgments

I want to thank members of my committee: Professor Clarence Barlow, Professor Curtis Roads and Professor Azer Akhmedov. I would also like to thank my father, Eugene Winter, for his support as well as Gregory Chaitin (whose work I was first introduced to by Curtis Roads), Larry Polansky and Lauren Pratt for their feedback and suggestions. Finally, I would like to thank the community of wonderful artists surrounding me. A proper list would be too long, but I think you know who you are...

Michael B. Winter
1026 South Santa Fe Avenue #203
Los Angeles, CA 90021
(current as of May 4th, 2010)

Curriculum Vitae

mwinter@unboundedpress.org
www.unboundedpress.org
(213) 446-4776

Education

University of California at Santa Barbara, Santa Barbara, CA
Ph.D. Candidate in Media Arts and Technology (May 2010) — 2007 – 2010
ABD (April 2008); Dissertation – *Structural Metrics: An Epistemology*

California Institute of the Arts, Valencia, CA
M.F.A. in Music Composition (May 2005) — 2003 – 2005

University of Oregon, Eugene, OR
B.S. with Music Technologies Emphasis (June 2003) — 2000 – 2003

Belmont University, Nashville, TN
B.M. Candidate in Commercial Music with Jazz Guitar Emphasis — 1999 – 2000

Teaching Experience

As Instructor or Assistant:

University of California at Santa Barbara, Santa Barbara, CA
Tranvergence – Teaching Assistant — Winter Term 2008
Assisted students in digital architecture, composition, and sound design implemented in Max/MSP/Jitter and other languages; held discussions/workshops on the philosophy of art and composition; assembled and maintained state of the art media laboratory, wrote networking software for motion capture/sound/video system
Digital Audio Signal Processing – Teaching Assistant — Fall Term 2007
Assisted students in digital audio signal processing, GUI design, and networking implemented in C/C++, and Java

University of Virginia, Charlottesville, VA
Technosonics – Teaching Assistant — Spring Term 2007
Taught basic music software such as Garage Band, sound design and synthesis implemented in Max/MSP, composition, and history of electronic music; evaluated student performance
Introduction to Music and Computers – Teaching Assistant — Fall Term 2006
Taught advanced music software such as Digital Performer, sound design and synthesis in Max/MSP, composition, and history of electronic music; evaluated student performance

California Institute of the Arts, Valencia, CA
Introduction to Electronic Media – Instructor — Spring Term 2004
Introduction to Electronic Media – Teaching Assistant — 2003 – 2005
Developed syllabus and course structure; taught advanced music software and programming languages for sound design and synthesis such as Max/MSP, composition, and electronic music history; managed media laboratory; evaluated student performance

As Guest Lecturer:

California Institute of the Arts, Valencia, CA
"Structural Metrics" – Guest Lecture — April 2009
On the epistemology of structural metrics

Columbia University, New York, NY
"On James Tenney's *Arbor Vitae* for String Quartet" – Guest Lecture — February 2008
As part of the "Tenney/Babbit Festival and Conference"

Teaching Experience (continued)

State University of New York, Buffalo, NY
"Lecture on the Machine and Music" – Guest Lecture — December 2006
On music, mathematics, and technology

State University of New York, Binghamton, NY
"Lecture on Infinity" – Guest Lecture — September 2005
On music with infinite realizations

Teachers, Mentors, and Influences

Clarence Barlow, Doug Barrett, Madison Brookshire, Michael Byron, Eric Clark, Nick Didkovsky, Tom Johnson, Joe Kudirka, Cat Lamb, György Ligeti, Alvin Lucier, James Orsher, Adam Overton, Michael Pisaro, Larry Polansky, Curtis Roads, Chiyoko Slavnics, Mark So, Jeffrey Stolet, James Tenney, Tashi Wada, Christian Wolff and Harris Wulfson – (this is not an all inclusive list)

Curatorial Experience

the wulf., Los Angeles, CA
Co-founder and Co-director — August 2008 – Present
Duties include managing all aspects of the 501(c)(3) non-profit arts organization including curating regular public events in experimental arts – www.thewulf.org

Industry Experience

Music Mastermind, Calabasas, CA
Java Media Programmer and Consultant — July 2008 – August 2010
Hibernate, Swing, MySQL, and Hidden Markov Models, JNI (MidiShare and JSyn), algorithmic music applications

Practical Expertise and Other Experience

Recording/Digital Audio Studio Use and Maintenance*
Advanced competence in studio design and maintenance; recording, engineering, editing, mixing, and mastering; sound design and synthesis

Java Programming Language*
Advanced competence in sound design, synthesis, signal processing, and algorithmic composition (JMSL, JScore, JSyn); creating graphics environments (Java2D); GUI development (Swing); basic competence in networking

C/C++ Programming Language
Basic competence in sound design, synthesis, signal processing, and GUI development

Lisp Programming Language
Basic competence in sound design, synthesis, signal processing

Max/MSP/Jitter Music and Video Programming Environment*
Advanced competence in sound design, synthesis, signal processing, and algorithmic composition; basic competence in digital video processing

Digital Audio Workstations – Pro-Tools and Digital Performer*
Advanced competence in recording, editing, mixing, and mastering

Music Notation Software – Finale*
Advanced competence in copying and engraving traditional and unconventional music scores

Referee
Peer reviewer for the Computer Music Journal

Practical Expertise and Experience (continued)

Freelance Copy Editor
Copied, edited, and engraved scores for composers including James Tenney, Larry Polansky, and Anne LeBaron; proofreader for the Directory of Recorded American Music at New World Records

Software Programmer
Designed transcription and algorithmic composition software in Java and Max/MSP for composers including James Tenney and Anne LeBaron

Guitar Instructor

Other Skills
Mathematica, Adobe and Macromedia Suites; Microsoft Office programs; web design and html

* Indicates expertise and teaching experience in this field

Selected Writings, Publications, and Presentations (including preprints)

- "LiveScore: Real-Time Notation in the Music of Harris Wulfson" (co-authored with G. D. Barrett) – forthcoming in the Contemporary Music Review
- "A Few More Words About James Tenney: Dissonant Counterpoint and Statistical Feedback" (co-author with L. Polansky and A. Barnett) – Submitted to the Journal of Mathematics and Music
- CD Review of *James Tenney Selected Works 1961–1969* – forthcoming in the Journal of the Society for American Music
- "Chordal and Timbral Morphologies Using Hamiltonian Cycles" (co-author with A. Akhmedov, Professor of Mathematics, University of California at Santa Barbara) – Preprint
- James Tenney Entry in Grove Encyclopedia (co-authored with S. Hanson, L. Polansky, and C. Streb)
- "On James Tenney's *Arbor Vitae* for String Quartet" – Contemporary Music Review, vol. 27, issue 1, 2008, London, England – Presented at Columbia University as part of the Tenney/Babbit Conference and Festival
- "Mavericks on Mavericks: James Tenney's Last Courses at CalArts" – MusikTexte, vol. 112, 2007, Berlin, Germany
- "Algorithmic Notation Generators" (co-authored with H. Wulfson and G. D. Barrett) – Presented at the 2007 New Interfaces for Musical Expression (NIME) Conference, New York, NY and presented at the 2007 International Computer Music Conference (ICMC), Copenhagen, Denmark
- "Lecture on the Machine and Music" – Presented in Dec. 2006 at the State University of New York, Buffalo
- "Lecture on Infinity" – Presented in Sept. 2005 at the State University of New York, Buffalo
- *Filter IV P.I.X.L. Study No. 1* – Sound recording of chamber piece with electronics, *DIY Canons*, CD, Pogus Records – 2005
- *The Other Self* – Electronic music piece premiered at the 2003 Society for Electroacoustic Music in the United States (SEAMUS) Convention, Tempe, AZ
- *Density Study No. 3* for 3 alto saxophones – 2002, Ruginenti Editore, Milan, Italy
- Scores of music compositions available on the web at www.unboundedpress.org

Residencies and Awards

- Scholarship and Central Fellowship – Department of Media Arts and Technology; 2009–2010; University of California at Santa Barbara;
- Residency – Atlantic Center for the Arts; May 2009; New Smyrna Beach, FL
- Scholarship – Department of Media Arts and Technology; 2007–2008; University of California at Santa Barbara;
- Scholarship and Fellowship – School of Music; 2006–2007; University of Virginia
- Scholarship – School of Music; 2003–2005; California Institute of the Arts
- Recognized as Outstanding Student in Electronic Music; 2003; University of Oregon

Residencies and Awards (continued)

- 1st Place – 2002 International Citta De Pavia Composition Competition; *Density Study No. 3* for 3 alto saxophones
- Scholarship – School of Music; 2001–2002; University of Oregon
- Scholarship – School of Music; 1999–2000; Belmont University

Primary Interests

- Music Composition
- Teaching
- Software design
- Art history and musicology (with emphasis on the American Experimental Tradition)
- Integration of art and technology
- Complexity theory
- Music theory
- Tuning theory
- Psychoacoustics, perception and cognition
- Mathematics and abstract topologies
- Physics
- Computer sciences and technologies

Languages

- English – fluent (native language)
- German – conversational

References

- Clarence Barlow, Professor of Music, University of California Santa Barbara, barlow@music.ucsb.edu
- Nick Didkovsky, Professor of Music, New York University, didkovn@mail.rockefeller.edu
- Anne LeBaron, Professor of Music, California Institute of the Arts, alebaron@calarts.edu
- Michael Pisaro, Professor of Music, California Institute of the Arts, mpisaro@calarts.edu
- Larry Polansky, Professor of Music, Dartmouth University, larry.polansky@dartmouth.edu
- Curtis Roads, Professor of Media Arts and Technologies, University of California Santa Barbara, clangtint@gmail.net

Brief Bio

- Michael Winter is an composer, curator, theorist, and software designer. He co-founded and co-directs (with composer eric km clark) *the wulf.*, a non-profit arts organization that presents experimental music free to the public in Los Angeles. Michael is a firm believer in art making as an experimental process and free information; e.g. open source code, free art, etc..

Miscellaneous Information

- Works list, performance list, press clippings, biography, sample syllabus, coursework, scores, recordings, and software can be viewed at **www.unboundedpress.org** or requested by email at **mwinter@unboundedpress.org**

Michael B. Winter
1026 South Santa Fe Avenue #203
Los Angeles, CA 90021
(current As of May 4th, 2010)

Works List

mwinter@unboundedpress.org
www.unboundedpress.org
(213) 446-4776

title	instrumentation
(2010)	
pedal, triangle machine, and (perhaps) coda	variable
after eons	variable
Approximating Omega	variable
(2009)	
piano machine	piano
recitation, code, and (perhaps) round	choir
field and perfect circuit	variable
for gregory chaitin	variable
for Sol LeWitt	variable
gray codes	variable
(2008)	
towards completeness	variable
small world	variable
dissection and field	variable
20 arrows 9 dashes	variable
for orin hildestad	variable
room and seams	variable
seams	variable
(2007)	
resonance i	variable
maximum change	variable
after a koan	solo violin
sound.sound	variable
many many for james orsher and peter kotik	variable

4 James Orsher	variable
Vein Transcription	variable
Entropic Canon	variable
Three	variable
Transplanting, 06.11.07-06.16.07 (or Transcription, USA!)	variable
a chance happening	solo piano
cactus for james orsher	variable
1 sample, x performers, y seconds	variable
4 Ascents for James Tenney	variable

(2006)	
Trajectories	variable
Prime Decomposition	variable
Intersections I	variable
Streams I	glissandi and electronics
diy for larry polansky	variable
three books and a dissertation	variable
in tone	variable
random I	variable
almost every piece	variable
for cassia streb	variable
for michael pisaro	variable
sort I	variable
nothing...I	variable

(2005)	
A Gaussian Canon	solo piano
Infinity III	variable
Lecture on Infinity	variable
a tone for Erik KM Clark	solo violin
A Flourish	variable
Infinity 2	variable

Perspectives I (revised 2010)	variable
Commas	variable
a set of pieces with one note	variable
(2004)	
Infinity 1	variable
Filter IV - P.I.X.L. Study No. 1	tam-tam, cellos, cymbals, voices and electronics
Filter III - Transformation Filter	strings and electronics
Chromatic Study	variable
Tri-Dimensional Canon	variable
Difference	2 clarinets
2 Filters	variable
Flux	voice, bassoon, bass, clarinet, didgeridoo, and/or... and 2176 sine tones
(2003)	
Fission (revised 2006)	flutes, clarinets, trumpet, horn, vibraphones, pianos, and strings
Coincidental Canon	orchestra
Telot's Crystal	fixed digital media
(2002)	
A Meditation for Solo Piano	solo piano
Beat Canon	variable
Density Study No. 2	fixed digital media
(2001)	
The Other Self	fixed digital media

Michael B. Winter
1026 South Santa Fe Avenue #203
Los Angeles, CA 90021
(current As of May 4th, 2010)

Performances
mwinter@unboundedpress.org
www.unboundedpress.org
(213) 446-4776

As artist

when	what	where	notes & miscellany
03.21.2010	*room and seams*	*The Bridge Progressive Arts Initiative; Charlottesville, VA*	as part of the second installation of Architectures of Sound, curated by David Kant and Cameron Hu
01.23.2010	*for Sol LeWitt*	*the wulf.*; Los Angeles, CA	Performed by the Red Light New Music ensemble.
01.13.2010	*Perspectives I*	Issue Project Room; Brooklyn, NY	As part of the MATA Interval Series – Architectures of Sound; performed by Casey Anderson, David Kant, Aaron Meicht and Phil Rodriguez.
11.24.2009	*a chance happening...*	Huddersfield University, UK	As part of the Huddersfield New Music Festival; performed by Philip Thomas.
11.19.2009	*for gregory chaitin*	The Stone; New York, NY	*the wulf.* @ The Stone.
10.24.2009	*field and perfect circuit*	Tenri Cultural Center; New York, NY	Performed by the Transit ensemble.
08.27.2009	*recitation, code, and (perhaps) round*	St. Wenceslas Church; Ostrava, CZ	As part of the 2009 Ostrava Days New Music Festival; performed by Canticum Ostrava.
08.19.2009	*field and perfect circuit*	Janacek Conservatory; Ostrava, CZ	As part of the 2009 Ostrava Days New Music Festival; performed by Joe Kudirka, Sam Sfirri, Taylan Susam, and David Kant.
06.11.2009	*for Sol LeWitt*	California Institute of the Arts; Valencia, CA	Performed by the Dogstar Orchestra.
06.05.2009	*for gregory chaitin*	Atlantic Center for the Arts; New Smyrna Beach, FL	As part of the Inside-Out Showcase.
05.28.2009	*for gregory chaitin*	Atlantic Center for the Arts; New Smyrna Beach, FL	
05.16.2009	*sort I*	*the wulf.*; Los Angeles, CA	Performed by the Sisters Streb.
05.01.2009	*towards completeness; dissection and field*	*the wulf.*; Los Angeles, CA	Performed by Beardman and Friends.
04.26.2009	*small world*	Huddersfield University, UK	As part of the Nothing New Conference; performed by the Edges Ensemble.

02.15.2009	*room and seams*	Willow Place Auditorium; Brooklyn, NY	Performed by the SEM ensemble.
10.11.2008	*sound.sound*	Circular Congregational Church; Charleston, SC	As part of the silence: 4'33" and beyond series curated by Jason Brogan.
09.11.2008	*20 arrows 9 dashes*	University of Nebraska, Omaha	As part of the Microscore Project @ ARTsaha! Festival; performed by Johnny Chang and Jessica CAtron.
09.09.2008	*a chance happening...*	*the wulf.*; Los Angeles, CA	Performed by Danny Holt.
09.01.2008	*20 arrows 9 dashes*	*the wulf.*; Los Angeles, CA	As part of the Microscore Project @ *the wulf.*; performed by Johnny Chang and Jessica CAtron.
06.14.2008	*for orin hildestad*	200 N. Ave. 57; Los Angeles, CA	As part of the Flag Day Experimental Barbecue; performed by Christa Graf, Orin Hildestad and Cassia Streb.
06.08.2008	*sound.sound*	30047 Madison Way; Val Verde, CA	Performed by the Dogstar Orchestra.
05.28.2008	*Rise I* from *4 Ascents for James Tenney*	The Pasadena Armory; Pasadena, CA	As part of the Los Angeles Microfest.
05.20.2008	*Telot's Crystal*	University of California, Santa Barbara	As part of the CreMATe Concert.
03.08.2008	*resonance i*	The Pescadrome; Santa Barbara, CA	Performed by James Orsher.
02.29.2008	*Transplanting 06.11.07-06.17.07*	The Pasadena Armory; Pasadena, CA	Performed by April Guthrie and Cassia Streb.
02.14.2008	*maximum change*	Willow Place Auditorium; Brooklyn, NY	Performed by the SEM ensemble.
11.13.2007	*after a koan*	University of California, Santa Barbara	Performed by Eric km Clark.
09.01.2007	*Streams I*	Janacek Conservatory; Ostrava, CZ	As part of the 2007 Ostrava Days New Music Festival; performed by the Ostravská Banda.
07.30.2007	*1 sample, x people, y seconds*	960 S. Oxford Ave. #312; Los Angeles, CA	As part of the Studio Apartment Series; performed by Orin Hildestad and Mark So.
07.13.2007	*a chance happening...*	Dangerous Curve; Los Angeles, CA	Performed by Danny Holt.
07.06.2007	*in tone*	862 Catalina St. #205; Los Angeles, CA	As part of the Studio Apartment Series; performed by Christa Graf, Orin Hildestad and Tashi Wada

07.06.2007	*three*	862 Catalina St. #205; Los Angeles, CA	As part of the Studio Apartment Series; performed by Eric km Clark, Christa Graf and Orin Hildestad.
06.09.2007	*Fission*	Riverside Church, New York, NY	Performed by the Locrian Players.
03.30.2007	*Trajectories*	University of Virginia, Charlottesville	Performed by the Now Ensemble.
12.01.2006	*Sort 1*	McGuffey arts Center; Charlottesville, VA	As part of the Noise in the System Series; performed by Wayla Chambo.
10.20.2006	*for michael pisaro*	California Institute of the Arts; Valencia, CA	As part of the Tender Buttons concert.
09.08.2006	*in tone; Sort I*	CINE Studio Space at the Artcraft Strauss Sign Factory; New York, NY	As part of the 1st annual concert of the Society of Automatic Music Notators; performed by SAMN.
03.22.2006	*lecture on infinity*	California Institute of the Arts; Valencia, CA	Performed by James Orsher.
03.15.2006	*lecture on infinity*	California Institute of the Arts; Valencia, CA	Performed by James Orsher.
11.17.2005	*A Flourish*	California Institute of the Arts; Valencia, CA	Performed by April Guthrie.
08.05.2005	*lecture on infinity*	SUNY; Binghamton, NY	
05.09.2005	*Commas*	REDCAT; Los Angeles, CA	Performed by the new century players.
05.04.2005	*theme and variatons from a set of pieces with one note*	California Institute of the Arts; Valencia, CA	As part of the Music for Six Pianos concert.
04.09.2005	*a set of piece with one note*	California Institute of the Arts; Valencia, CA	As part of A Concert of One Note Pieces.
03.20.2005	*Filter III*	California Institute of the Arts; Valencia, CA	Performed by Cassia Streb.
01.28.2005	*Flux*	REDCAT; Los Angeles, CA	As part of the CEAIT Festival; performed by Thadeus Frazier Reed.
05.03.2004	*Fission*	California Institute of the Arts; Valencia, CA	Performed by the New Century Players.
04.30.2004	*Filter I* from *2 Filters*	California Institute of the Arts; Valencia, CA	As part of the new music for winds and friends concert; performed by the CalArts woodwind ensemble.

04.24.2004	*Chromatic Study*	California Institute of the Arts; Valencia, CA	As part of the Quiet Night/Quiet Music concert; performed by Mark So.
04.16.2004	*Filter III*	California Institute of the Arts; Valencia, CA	Performed by Mark Menzies.
03.04.2005	*Flux*	California Institute of the Arts; Valencia, CA	Performed by Thadeus Frazier Reed.
02.09.2004	*A Meditation For Solo Piano*	California Institute of the Arts; Valencia, CA	Performed by Vicki Ray.
11.06.2003	*The Other Self*	Stanford University; Palo Alto, CA	At the Center for Computer Research in Music and Acoustics
05.31.2003	*Telot's Crystal*	University of Oregon, Eugene	As part of Future Music Oregon.
03.14.2003	*The Other Self*	Arizona State University; Tempe, AZ	As part of the 2003 SEAMUS conference.
06.01.2002	*Density Study No. 2*	University of Oregon, Eugene	As part of Future Music Oregon.
11.17.2001	*The Other Self*	University of Oregon, Eugene	As part of Future Music Oregon.

As performer

when	what	where	notes & miscellany
02.06.2010	*Piker* and *freeHorn* by Larry Polansky	*the wulf.*; los angeles, ca	
01.13.2010	*A Few Rooms* by G. Douglas Barrett	Issue Project Room; New York, NY	As part of the MATA Interval Series – Architectures of Sound.
01.11.2010	*The Collection* by Michael Pisaro	Zipper Hall; Los Angeles, CA	As part of the Monday Evening Concert series.
11.19.2009	*LiveScore* by Harris Wulfson; *JOHN ASHBERY (2 short litanies)* and *readings 32 - Landscapeople* by Mark So; *Lucifer in the Shadowlands* and *Elevator Music* by Laura Steenberge	The Stone; New York, NY	*the wulf.* @ The Stone.
11.07.2009	*harmony #* by David Kant	*the wulf.*; Los Angeles, CA	
10.16.2009	*Distance Learning* by John Lely	*the wulf.*; Los Angeles, CA	
10.04.2009	*LiveScore* and *Durations* by Harris Wulfson	*the wulf.*; Los Angeles, CA	

08.19.2009	*beauty and industry* by Joe Kudirka	Janacek Conservatory; Ostrava, CZ	As part of the 2009 Ostrava Days New Music festival.
07.05.2009	*ordinary education (the true course of events)* by Mark So	REDCAT; Los Angeles, CA	
06.12.2009	*for Dan Flavin* and *...the sharpening line of crests* by Sam Sfirri	*the wulf.*; Los Angeles, CA	
06.11.2009	*huddersfield trio* by Joe Kudirka; *for maaike schoorel* Taylan Susam	California Institute of the Arts; Valencia, CA	With the Dogstar Orchestra.
06.09.2009	*composition 1960 #7* by La Monte Young; *swell piece no. 2* by James Tenney; *the void* by Ulrich Krieger; *ascending series no. 5* by Michael Pisaro; *assembly (melody 1)* by Casey Thomas Anderson; *bands* by John Hastings	California Institute of the Arts; Valencia, CA	With the Dogstar Orchestra.
01.23.2009	*one for all and all for one* - the music of Sara Roberts	*the wulf.*; Los Angeles, CA	As part of A Program of Participatory Pieces.
01.10.2009	*westin bonaventure hotel: documentary music #3* by James Orsher	*the wulf.*; Los Angeles, CA	
12.07.2008	*In a Large Open Space* by James Tenney; *fades/crossfades* by Casey Thomas Anderson	*the wulf.*; Los Angeles, CA	
10.06.2008	*A Few Silences* by G. Doug Barrett; *State of the Union* by Adam Fong	downtown; long beach, CA	As part of Soundwalk08.
09.06.2008	*canon*, *beauty and industry* and *solidarity* by Joe Kudirka; *two durations*, *excersize*, *three yellow events* by Goerge Brecht	*the wulf.*; Los Angeles, CA	
08.22.2008	*durations* by Harris Wulfson; *unfurl* by Alison Knowles	*the wulf.*; Los Angeles, CA	
06.12.2008	*the rain of alphabets* by Michael Pisaro	California Institute of the Arts; Valencia, CA	With the Dogstar Orchestra.
06.08.2008	*Motor Vehicle Sundown (Event)* by George Brecht; *pleasure cruise for ... (series)* by Mari Garrett	30047 Madison Way; Val Verde, CA	With the Dogstar Orchestra.

06.06.2008	*Fragments* by Jason Brogan; *Jesus' Blood Never Failed Me Yet* by Gavin Bryars; *Chair Piece for George Brecht* by Alison Knowles; *What Do We Do Now? Performance* by Adam Overton	California Institute of the Arts; Valencia, CA	With the Dogstar Orchestra.
06.04.2008	*weiss/weisslich 5b* by Peter Ablinger	California Institute of the Arts; Valencia, CA	With the Dogstar Orchestra.
06.03.2008	*Joseph Kudirka* by Mark So; *Swell Piece* by Kames Tenney; *2008(3)* by Manfred Werder	California Institute of the Arts; Valencia, CA	With the Dogstar Orchestra.
04.25.2008	*the mayan empire* by Mark So	The Pasadena Armory; Pasadena, CA	As part of the Los Angeles Microfest.
03.21.2008	*The Collection* by Michael Pisaro; *for percussion perhaps, or ...* by James Tenney; *ihwetunings for twenty* by Antoine Beuger; *4'33"* by John Cage	REDCAT; Los Angeles, CA	As part of the 4'33" and Beyond concert; with the Experimental Music Workshop.
03.08.2008	*doublings (6)* by James Orsher	University of California, Santa Barbara	
03.03.2008	*the delhi train station* by James Orsher	Sea and Space Gallery; Los Angeles, CA	As part of The Freed Reed concert.
02.25.2008	*Present Joys* by James Orsher	University of California, Santa Barbara	
11.11.2007	*just walking around [ashbery series]* by Mark So	Goleta County Beach Park; Goleta, CA	As part of the Mark So: Late Early Works festival.
11.10.2007	*sigh of our present (blue sonata) [ashbery series]* and *readings 20 - syringa [ashbery series]* by Mark So	Downtown Santa Barbara Farmers' Market; CA	As part of the Mark So: Late Early Works festival.
11.10.2007	*when the sun went down [ashbery series]* by Mark So	University of California, Santa Barbara	As part of the Mark So: Late Early Works festival.
11.09.2007	*new leaves (a mood survives) [ashbery series]* by Mark So	University of California, Santa Barbara	As part of the Mark So: Late Early Works festival.
11.05.2007	*the roman empire* by Mark So	University of California, Santa Barbara	As part of the Mark So: Late Early Works festival.
10.06.2007	*some tunes volumes I, II and III* by Eva-Maria Houben	University of California, Santa Barbara	

08.31.2007	*second symphony* by John Lely; *composition 1960 #7* by La Monte Young; works of Ben Patterson.	Ostrava Cultural Center; Ostrava, CZ	As part of the 2007 Ostrava Days New Music Festival; performed with Fluxus and Associates.
08.15.2007	*Exercise 1* by Christian Wolff	Hotel Maria; Ostrava, CZ	As part of the 2007 Ostrava Days New Music Festival.
07.30.2007	*swan vestas (cutters choice)* by Johnny Chang; *sigh of out present (blue sonata) [ashbery series]* by Mark So	960 S. Oxford Ave. #312; Los Angeles, CA	As part of the Studio Apartment Series.
05.09.2005	*Ancient Music* by Eric km Clark	REDCAT; Los Angeles, CA	With the New Century Players.
05.04.2005	*bad pianos work best* by Joe Kudirka	California Institute of the Arts; Valencia, CA	As part of the Music for Six Pianos concert.
04.11.2005	*Ordinary Matter*, *Fits and Starts* and *Changing the System* by Christian Wolff	California Institute of the Arts; Valencia, CA	With the Experimental Music Workshop.
09.08.2004	*chamber music (for george brecht)*, *(night) for percussion perhaps, or …*, *the swell pieces (1, 2, and 3)*, *harmonium # 7* and *in a large, open space* by James Jenney	California Institute of the Arts; Valencia, CA	With the Experimental Music Workshop.

Abstract

Structural Metrics: An Epistemology

by

Michael Benjamin Winter

Structural Metrics is an epistemological survey of methods of comparing structures. The text shows how fundamental limits of knowledge affect our ability to objectively compare the intrinsic rules and relationships that govern two given objects. Several ways of analyzing, generating, comparing and mutating structures are examined throughout bringing four fields together: music, communication theory, algorithmic information theory and graph theory. Even though *Structural Metrics* is rooted in music, it applies to many other fields and is quintessentially pandisciplinary.

The epistemological survey results in two important and novel findings. The first finding came from asking whether it is possible to objectively determine structural similarity of two objects (such as pieces of music). What are the limits of such metrics? A survey of structural metrics reveals a fundamental tradeoff of objectivity for practicality in structural analysis. While we may know that two objects share information, it is basically impossible to know exactly how much information they share. The most objective metric is based on the minimal programs of two objects. A minimal program, defined by Gregory Chaitin as the smallest computer program that calculates a given

object, is a maximally compressed encoding of an object. Knowing whether or not a program is minimal is essentially impossible due to the halting problem, which is the inability to compute whether an arbitrary program halts. The halting problem was first pointed out by Alan Turing in 1936. That is, what we need to know to objectively determine differences in structure, we cannot know. All other methods of structural analysis that do not use maximally compressed representations of the objects under examination sacrifice the objectivity of the analysis. What is perhaps often gained are more practically efficient methods. The second finding resulted from the initial goal of implementing structural metrics in evolutionary algorithms. Practical structural mutations cannot be made to the genome of an object (the computer program that generates it) because a minor mutation may result in a non-halting program. This raised the question of how an abstract mathematical theory of evolution can incorporate mutations to the genome and model 'real-world' properties of evolution (regardless of practicality). The result is a new abstract mathematical theory of evolution that takes into account the halting problem as the reason for evolutionary advancement in both minor and major evolutionary leaps.

The text is organized as follows. First, preliminary discussions on music, music related philosophies and information theory are provided. Then, ways to generate and represent musical structures are illustrated followed by a survey of several structural metrics. Finally, it is shown how these metrics can be implemented in 'artificial' evolutionary algorithms in order to reveal how such algorithms do not mimic several important properties of 'real-world' evolution. Thus, the text itself has a trajectory from the generative to the analytical and back. In conclusion, a new abstract mathematical

definition of 'real-world' evolution is given along with a return to the more philosophical discussions set forth in the beginning. At the end, I present several musical works that informed and were informed by the research.

Contents

List of Figures **xxvi**

List of Tables **xxvii**

Preface **xxviii**

1 Introduction **1**
1.1 Genesis of *Structural Metrics* 1
1.2 The World is Programmable: An Introduction to Digital Philosophy. . . 6
1.3 A Note About Program-Size Versus Run-Time 9
1.4 Music, Information, Communication and Function 9
1.5 Measures of Structure; Analysis and Empiricism 11
1.6 Encoding Structure 12
1.7 A Survey of a Few Methods of Musical Analysis with Respect to Program-Size Complexity 13
1.8 What is to Come 16

2 Structure **17**
2.1 Structure and its Analogs: Local and Global Morphological Constraints 17
2.2 Apriority and Posteriority 20
2.3 Historical Premise 22
2.3.1 Inexplicit Examples 22
2.3.2 Explicit Examples 25
2.4 A Few More Thoughts on Structure 27

3 Metrics **29**
3.1 Mathematical Definition 29
3.2 Examples of Metrics 30
3.2.1 The Euclidean and Taxicab Metrics 30
3.2.2 Harmonic Distance 31
3.2.3 Morphological Metrics 32
3.2.4 Information-Based Metrics 34

4 Structural Metrics **36**
4.1 The Tradeoff: Objectivity for Practicality . . . 36
4.2 Mutual Information . . . 37
4.3 The Most Objective Metric: Mutual Information of Minimal Programs . 38
4.4 Normality: A Metric that Accounts for the Distribution of Substrings. . 42
4.5 Graph Comparison . . . 47
4.5.1 The Leap Into the Truly Subjective . . . 47
4.5.2 Extensions . . . 49
4.5.3 Graph Representations and Correlate Metrics . . . 49
4.6 A Few More Well-Studied Ideas:
Compression Schemes, Cross Correlation and Some Others . . . 52
4.7 Perhaps a Few Tricks for Analyzing Music . . . 54

5 Structural Mutations **55**
5.1 What is Being Mutated? . . . 55
5.2 Untargeted and Targeted Mutations with Controlled Trajectories . . . 57
5.3 The General Algorithm . . . 57
5.3.1 Untargeted . . . 57
5.3.2 Targeted . . . 58
5.4 Closer to Life: Problem Solving Algorithms . . . 59
5.5 A Stepping Stone to a Mathematical Model of Evolution . . . 60

6 Conclusion **61**
6.1 Recasting a Mathematical Definition of Evolution . . . 61
6.1.1 Examining Metabiology . . . 61
6.1.2 Growing an Oracle: A New Mathematical Theory of Evolution . 64
6.2 Important Findings . . . 65
6.3 Future Work and An Open Question . . . 67
6.4 A Return to Philosophies on Music: Computation and Composition . . 69
6.4.1 Ephemera . . . 69
6.4.2 Scalability and Relativity . . . 71
6.4.3 On Documentation . . . 73
6.4.4 A Final Thought . . . 74

Bibliography **75**

A Scores **87**
A.1 *maximum change* . . . 88
A.2 *room and seams* . . . 96
A.3 *dissection and field* . . . 104
A.4 *small world* . . . 116
A.5 *towards completeness* . . . 123
A.6 *for Sol Lewitt* . . . 132
A.7 *for gregory chaitin* . . . 135
A.8 *field and perfect circuit* . . . 137
A.9 *recitation, code, and (perhaps) round* . . . 140

A.10 *Approximating Omega* 145
A.11 *pedal, triangle machine, and (perhaps) coda* 172

List of Figures

2.1 A directed graph 19
2.2 An undirected graph 19
2.3 A labeled graph 20
2.4 An icosehedron with a Hamiltonian cycle 21
2.5 *For 1, 2 or 3 People* excerpt of instructions 22
2.6 *Psaltery* sketch 23
2.7 Harmonic lattice of *Arbor Vitae* 24
2.8 *Trio* graph (from cover of score) 26
2.9 The canonical transition sequence for an 8-bit Beckett-Gray code 27

3.1 An illustration comparing the Euclidean and taxicab metrics 31

6.1 An instruction from Sol LeWitt's *Wall Drawing 305* (1977) 72
6.2 Score of James Tenney's *HAVING NEVER WRITTEN A NOTE FOR PERCUSSION* (1971) 73

A.1 Graph of $T_{3,2,0}$ with a Hamiltonian cycle 89
A.2 de Bruijn graph of $B(3,2)$ 96
A.3 *for Sol LeWitt*: illustration of instructions 132
A.4 *for Sol LeWitt*: 4 visual realizations 133

List of Tables

3.1 The categories of Polansky's morphological metrics 33

Preface

> "...the equivocal paradigm of all fidelity: proofs of love, ethical rigor, the coherency of a work of art, the accordance of a politics with the principles which it claims as its own—the exigency of such a fidelity is propagated everywhere: to be commensurable to the strictly implacable fidelity that rules the discourse on being itself. But one can only fail to satisfy such an exigency; because the fact that it is the type of connection which is maintained in the mathematical text—despite it being indifferent to the matter—is something which proceeds directly from being itself. What one must be able to require of oneself, at the right time, is rather that capacity for adventure to which ontology testifies, in the heart of its transparent rationality, by its recourse to the procedure of the absurd; a detour in which the extension of their solidity may be restituted to the equivalences: 'He shatters his own happiness, his excess of happiness, and to the Element which magnified it, he rends, but purer, what he possessed.'"
>
> Alain Badiou from *Being and Event* [14]

This preface provides a philosophical context to what follows with three highly related premises that warrant initial address.

1) Music and mathematics exist for different reasons, yet objects in one domain can derive objects in the other. While each is not requisite for the other, as Alain Badiou expresses, their connection "is maintained in the mathematical text, despite it being indifferent to the matter."

Tom Johnson also reflects on this connection, stating:

> "This Marcel Duchamp principle, the 'readymade' or the 'objet trouvé,' is now recognized everywhere as a perfectly valid way of making art and it is

> now quite natural that a composer or artist might choose to work with a found mathematical object, like Pascal's triangle or the Narayana series or some automaton, just as well as with a urinal, a bicycle wheel, a comb, or a bottle rack." [64]

With respect to analytical music theory, *Structural Metrics* examines ways to develop mathematical objects that represent musical structures. With respect to generative music theory, *Structural Metrics* examines ways to develop musical structures from mathematical objects and more importantly, to find mathematical objects for music derivation.

2) Music is sets of possibilities and various sets may share information. A musical process is the genesis of a musical set over time encompassing all events from concept to percept with respect to sound and silence. This includes instruction, interpretation and realization. In this context, the definition of 'event' is adopted from Badiou: it is a rupture from the status quo, which illuminates the space of all possible events. That is, the space of all events that have happened and the void of happenings that may yet come. This illumination, by virtue, perturbs the space. An event is the only happening that actually truly defines the world—the set of possibilities—being explored. While one can attempt to define a set of possibilities from the present looking forward, the actual possibilities of a concept are only illuminated in the presence of the event looking backwards.

A simple iteration of one (of many) musical processes illustrates this. A composer has a concept, which may already define several possibilities (especially if it involves randomness, stochastics and/or indeterminacy). S/he creates a score which is a set of instructions—written, verbal, machine, etc.—that accounts for the above possi-

bilities and functions as a prompt to action [16]. (Note that several scores can produce similar results [71].) An interpreter, be it a performer or a computer, then realizes the score into sounds and silences. These sounds and silences are the piece—nothing else. The piece is then experienced/perceived by a listener and considered. The formation of the concept, the creation of a score that outlines possibilities, the interpretation and realization, and finally experience/perception are all events. They are happenings manifested from the space of all events. Each step away from the concept may illuminate more and more possibilities while ultimately leading to a series of singular events.

If music is set, then possible realizations of a given piece might be super/subsets of or intersect with the possible realizations of some other piece. More importantly, these sets may share information. It may take less to calculate or describe them together than separately. *Structural Metrics* is an epistemology—the exploration of inherent limits of knowledge—of trying to measure the mutual information of two objects and thus extends to the mutual information of pieces of music.

3) At once, music can be logical and absurd. A piece may generally adhere to a set of axioms/truths, but may always break from those logics to establish opposing logics (or absurdities). Furthermore, absurdities may result from the incalculability of the concept-to-percept transparency of music, which is the inability to know the extent that a perceiver will understand the logic of a work's concept through musical experience. This is partially due to what James Tenney calls the subjective set:

> "...[the] expectations or anticipations [arising during a musical experience] previous to those occasioned by the particular piece of music now being considered."
>
> [99]

The experience of music can be transcendent. Any logical rigor of a piece may or may not be perceived and/or appreciated as such. A perceiver need not understand the rules that govern a piece of music in order to enjoy it. Various logics provide a means to discuss music. Composers have always attempted to generate and understand music through logic, especially using mathematical formalizations. The writings of Charles Ames [6, 5, 8, 7, 9, 10, 11], Clarence Barlow [15], John Cage [21], Tom Johnson [62, 63, 64, 65, 66, 67], Larry Polansky [84, 82], James Tenney [98, 100] and Iannis Xenakis [106], among many others, provide an abundant wealth of theoretical insight.

James Tenney posits the following bases for a theory of music:

> "First, it should be descriptive—not pre- (or pro-) scriptive—and thus aesthetically neutral. That is, it would not presume to tell a composer what should or should not be done, but rather what the results might be if a given thing is done. Second, it should be culturally/stylistically general... Finally—in order that such a theory might qualify as a 'theory' at all,... it should be (whenever and to the maximum extent possible) quantitative. Unless the propositions, deductions, and predictions of the theory are formulated quantitatively, there is no way to verify the theory and thus no basis for comparison with other theoretical systems." [98]

I believe very strongly in Tenney's bases for a comparative theory of music. Comparability is necessary for the epistemological survey that follows. Still, I acknowledge that some theories (such as certain theories of aesthetics) may not be quantifiable at all. I believe the more important point Tenney is making is that when a theorist suggests that one "should or should not" do something, s/he is in fact assuming the authority to tell others what is right or wrong or good or bad with respect to music. The quality of music is subjective and varies from person to person according to their personal taste. Discourse that suggests what "should or should not be done" is in fact

not a theory at all, be it quantitative or aesthetic, but rather just an expression of personal taste.

The theories formulated by the composers above provide several suitable models for the following epistemology of comparing structures as they tend to subscribe to Tenney's bases. While these theories are quantitative, they are predicated on subjective observation (as with almost all theories). Henri Poincaré wrote in his essay "On the Foundations of Geometry" the following:

> "We choose this geometry rather than that geometry, not because it is more true, but because it is more convenient... If our experiences should be considerably different, the geometry of Euclid would no longer suffice to represent them conveniently, and we should choose a different geometry... In fine, it is our mind that furnishes a category for nature. But this category is not a bed of Procrustes into which we violently force nature, mutilating her as our needs require. We offer nature a choice of beds among which we choose the couch best suited to her stature." [81]

Though the usability of a theory might substantiate it to some extent, in music, experiences are often "considerably different," especially from person to person. Experiences are also independent of the logic behind the concept of a work. Just as there are several ways to create music, there are several ways to experience and perceive music. As Cage points out, "composing's one thing, performing's another, listening's a third" [21]. Because of this, it is difficult to prove that a particular means of analysis is objective and comprehensive. Even though mathematics can aid in formalizing an analytical method, the concepts of formal axiomatic systems and rigorous proofs are rather specific to mathematics with no real equivalent in music. Furthermore, as Kurt Gödel, Alan Turing and Gregory Chaitin have all shown, mathematics cannot even fully prove itself [27, 50, 101]. However, ideas in different fields—music, science, mathematics,

philosophy, etc.—may have profound implications on one another that may engender hitherto unheard musics, mathematics, and philosophies.

The statement that 'at once, music can be logical and absurd' is illustratable in many ways. Perhaps one beauty of music is that musicians, and artists in general, often freely combine determinism and indeterminism, symmetry and asymmetry, dynamism and stasis, and many other opposing logics. Also, people share music throughout the world even though it has little to no materiality (this is certainly the case if taken literally). Music is one of the few acts of humankind where the absurd may seem logical and the logical may seem absurd. In the quote preceding this chapter, Badiou states, "What one must be able to require of oneself, at the right time, is rather that capacity for adventure to which ontology testifies, in the heart of its transparent rationality, by its recourse to the procedure of the absurd..." The transparent rationality of music is like ontology itself. Understanding the fidelity of music will sometimes require us to recourse to the procedure of the absurd.

It is with these fundamental premises that we proceed...

Chapter 1

Introduction

> "...(structure) usually refers more to an internal aspect of sound—'connections' or interrelations among component parts which (interrelations) are not necessarily apparent 'on the surface' of the form—i.e. in its shape."
>
> James Tenney from *META + HODOS* [99]

> "You have a law of nature if there is compression, if the experimental data is compressed into a computer program that has a smaller number of bits than are in the data that it explains. The greater the degree of compression, the better the law, the more you understand the data. But if the experimental data cannot be compressed..., if the smallest program for calculating it is just as large as it is then the data is lawless, unstructured..."
>
> Gregory Chaitin from *Meta Math!* [33]

1.1 Genesis of *Structural Metrics*

Initially, I started out with a rather simple goal: to define a set of viable (perhaps even practical), objective methods to compare structures of various pieces of music with the hope of extending these methods to evolving musical materials and comparing structures in other domains. First, structure needed an unequivocal definition. Two statements on structure are provided at the beginning of this chapter: one on the inter-

relations of elements and the other on compressibility (or redundancy). I adopted these as the primary criterion of structure.

In *META + HODOS*, James Tenney [99] describes two primary aspects of form: shape and structure. He explains shape as an "external" aspect of sound based on parametric profiles, and structure as an "internal" aspect of sound based on the connectivity of elements. In "Morphological Metrics," Larry Polansky provides several methods of comparing morphologies (or shapes defined by ordered-lists of elements). *Structural Metrics* is the sister work of Polansky's seminal article investigating metrics on the other aspect of form; i.e. structures as opposed to shapes.

While this dissertation has changed significantly from my initial investigations, I am borrowing to some extent (as I originally set out to do) the format of "Morphological Metrics" along with certain principles such as "aesthetic neutralism" as well as "comprehensiveness" with respect to musical material and what Tenney refers to as "scale," which is the perceptual level (or scope) one is investigating/considering. Temporal gestalt analysis, morphological metrics, and structural metrics apply to any materials, any aesthetic, between and within any "hier/hol/heterarchical level" [82], and between or within any two objects. In *META + HODOS*, the element-clang-sequence hierarchy illustrates this. A sequence on one level can be an element on another, higher level. In "Morphological Metrics," morphologies or morphological metrics on one level can be elements of a morphology on another, higher level. Finally, in the theory of *Structural Metrics*, structures or structural metrics on one level can be elements of another, higher-level structure.

Structural analysis in any field is nothing new. However, the discourse on comparing structures (especially in music) often lacks a concrete definition of structure and unequivocal gauges of the objectivity (or conversely, subjectivity) and even accuracy of given methods. I thought graph theory may provide solutions to these problems. My initial research investigated the application of graph theory to the analysis and generation of musical structures. I developed and formulated this approach over many years, but still could not comprehensively address or conquer any of the problems inherent to other techniques. These methods, which are still included in this text, while perhaps novel with respect to music theory, turned out to have the same shortcomings of former methods of structural analysis. Essentially, I became stuck and needed to figure out why the above problems did not go away. In lieu of this, I changed my focus from practical applications of structural metrics to an epistemology of structural metrics in order to determine why objective structural analysis seemed so difficult. *Structural Metrics* is an epistemological survey of methods of comparing structures. The text shows how fundamental limits of knowledge affect our ability to objectively compare the intrinsic rules and relationships that govern two given objects.

The epistemological survey results in two important and novel findings. The first finding came from asking whether it is possible to objectively determine structural similarity of two objects (such as pieces of music). What are the limits of such metrics? A survey of structural metrics reveals a fundamental tradeoff of objectivity for practicality in structural analysis. While we may know that two objects share information, it is basically impossible to know exactly how much information they share. Algorithmic information theory provided the benchmark of a completely objective (however terribly

impractical) structural metric that illuminates clear limits of knowledge when comparing structures. The metric is based on the minimal programs of two objects. A minimal program, defined by Gregory Chaitin as the smallest computer program that calculates a given object [27], is a maximally compressed encoding of an object. Knowing whether or not a program is minimal is essentially impossible due to the halting problem, which is the inability to compute whether an arbitrary program halts. The halting problem was first pointed out by Alan Turing in 1936 [101]. That is, what we need to know to objectively determine differences in structure, we cannot know. All other methods of structural analysis that do not use maximally compressed representations of the objects under examination sacrifice the objectivity of the analysis. What is perhaps often gained are more practically efficient methods. The second finding resulted from the initial goal of implementing structural metrics in evolutionary algorithms. Practical structural mutations cannot be made to the genome of an object (the computer program that generates it) because a minor mutation may result in a non-halting program. This raised the question of how an abstract mathematical theory of evolution can incorporate mutations to the genome and model 'real-world' properties of evolution (regardless of practicality). The result is a new abstract mathematical theory of evolution that takes into account the halting problem as the reason for evolutionary advancement in both minor and major evolutionary leaps. The limits of knowledge explored through structural metrics can actually be integrated into a rather clear (and simple) mathematical definition of evolution that builds upon earlier theories.

Throughout the text, methods of analyzing, generating, comparing and mutating structures are formalized from an already established discourse and are examined

in such a way that shows the objectivity and accuracy of the various methods. Four fields are brought together to achieve this: music, communication theory, algorithmic information theory and graph theory. While some of the methods of structural metrics herein are not at all practical, they help show the limits of our understanding of structure. Even though *Structural Metrics* is rooted in music, as will be shown throughout, it applies to many other fields and is quintessentially pandisciplinary.

The following text is organized as follows. In this chapter, preliminaries for the following discussion are provided. Then, ways to generate and represent musical structures are illustrated followed by the examination of several structural metrics. Finally, it is shown how these metrics can be implemented in 'artificial' evolutionary algorithms in order to reveal how such algorithms do not mimic several important properties of 'real-world' evolution. Thus, the text itself has a trajectory from the generative to the analytical and back. In conclusion, a new abstract mathematical definition of 'real-world' evolution is given along with a return to the more philosophical discussion set forth in the preface. In an appendix at the end, I present several musical works that informed and were informed by the research.

Ultimately, one of my hopes is that readers will explore and discover new methods of analysis and composition by examining and developing the following ideas, extending and altering them as they see fit.

1.2 The World is Programmable: An Introduction to Digital Philosophy.

It is conceivable that our world is completely digital, made up entirely of bits of information. Strong proponents of this idea, often called "digital philosophy" or "digital physics," include Gregory Chaitin, Seth Lloyd and Stephen Wolfram (see [26, 76, 105]). In "Programming the Universe" [76], Lloyd suggests that the world around us might be created by a quantum computer that is the universe. In order to proceed with our epistemological survey, we must make this one assumption that allows us to equate the fundamental limits of knowledge with the fundamental limits of computation.

Digital philosophy is strongly rooted in Alan Turing's conception of an abstract computing machine [101]. These "Turing machines" are essentially theoretical computers, which became the blueprint of the modern day computer. Furthermore, there are "universal Turing machines" that can compute anything computable. Turing showed that these machines often fail to halt and output a final computation. Most of them get into a loop that continues endlessly. One of Turing's most important findings is that there is no way of computing whether an arbitrary Turing machine will halt. The question can be stated as follows: 'given an arbitrary computer program, is there another computer program that will determine whether or not the input computer program will halt?' The answer is no. This result, known as the "halting problem," propagates itself everywhere when trying to compare structures.

First, it may be helpful to spell out a few examples borrowed and extended from Chaitin:

$$\text{binary input} \longrightarrow \text{computer} \longrightarrow \text{binary output}$$

$$\text{theory} \longrightarrow \text{computer} \longrightarrow \text{data}$$

$$\text{concept} \longrightarrow \text{computer} \longrightarrow \text{piece of music}$$

$$\text{DNA (genotype)} \longrightarrow \text{computer} \longrightarrow \text{organism (phenotype)}$$

All of these are analogous in a computational sense. The data on the left is a computer program and the data on the right is the output of the given computation. Put otherwise, the data on the left is an encoding of structure and the data on the right is the artifact or the morphology. We distinguish structure from morphology as follows. Morphology, which is an ordered list of elements, is a completely uncompressed representation of an object. Structure is the set of rules and relationships between individual elements and groups of elements that govern the morphology.

The model in the examples above is not necessarily enough for objective structural analysis. On the left side of each of these equations, one really needs the smallest (most elegant) program that computes the given output. A minimal/elegant program is maximally compressed. Knowing the minimal program ensures that extraneous information has not been added to the encoding. A problem with subjective structural encodings is that, most likely, more information than is necessary is used to encode the structure. An elegant and objective theory explains the data in the smallest number of bits possible. Also, as Chaitin explains in the quote at the beginning of this chapter, if

the minimal program that outputs a given object is about the same size as the output itself, then the object "is lawless, unstructured."

The next question that begs to be asked is 'how does one go from the chicken to the egg?' How can one find a minimal program given a certain output? Chaitin shows that this is essentially impossible [27] . The proof (by contradiction) goes as follows:

Given a formal axiomatic system, create a paradoxical program P that generates all theorems in that system until it finds a proof that a particular program Q (which is larger in bits than the formal axiomatic system) is elegant. So P must be smaller than Q and at the same time be able to find Q. Such a P is a contradiction because if Q is elegant, there cannot be a program smaller than Q that generates it.

This turns out to be a really dangerous version of incompleteness first proved by Kurt Gödel [50]. It is the reason why going from the chicken to the egg—from the object to what created it—is often impossible. One can attempt to run all possible computer programs of size less than the desired output, but it is impossible to know whether or not these smaller (possibly minimal) programs will halt in ten, one hundred, one thousand, one million years, etc., or if they will just run forever. You might find a program smaller than the morphology it outputs, but if even smaller programs that have not yet halted continue running in the background, you cannot know if the relatively small program that you found is elegant.

1.3 A Note About Program-Size Versus Run-Time

Tenney also points out in *META Meta+Hodos* [99] that morphology/shape is time-dependent while structure is an out of time characteristic. This fact is recapitulated in the computation of elegant programs. An elegant program is often smaller (in bits) than the data it outputs. If so, the number of computations (or run-time) of the program might be greater than the size of the output. The creative musical process also indicates this: composers typically do not spontaneously generate music, rather they set out with an initial (often simple) concept that develops over time. For the most part, the amount of time taken to create a piece (including all thought processes) generally exceeds the length of time of that particular piece.

Note that we search for minimal programs strictly for analytical reasons. Minimal programs provide an ideal for comparing structures. Nonetheless, it is certainly possible that art might adhere to a different paradigm such as the computations of very large computer programs that have long run-times with simple outputs. That is, a computation where each bit of the input program contributes a fractionally small amount of information to the output.

1.4 Music, Information, Communication and Function

The music-making process is a series of communications through several media (ultimately resulting in sound and silence). Therefore, principles of communication set forth in Claude Shannon's seminal work, "A Mathematical Theory of Communication" [92] can apply to music: the composer encodes information (a message) and transmits

it through some channel (like performers vibrating air); then, a listener receives and decodes the information. Tenney explicitly shows how Shannon information theory can apply to music in *META Meta+Hodos* [99]. Also, Abraham Moles has written significantly about information theory as it applies to perception and aesthetics (for example, see [77]).

Typically, one intends unambiguous communications without any 'noise in the system.' The sender and the receiver both know the codewords and share the same grammar. Perhaps what distinguishes art is that it does not abide by such standards. The composer may encode a message that already contains ambiguity (openness), it may be further obfuscated by interpretation and then the listener most likely does not have the same dictionary and/or grammar as the composer when decoding the message. These factors contribute to the incalculability of concept-to-percept transparency explained in the preface.

In "Cognitive Constraints on Compositional Systems," Fred Lerdahl [72] points out that the set of compositional grammars differs from the set of listening grammars. The intersection of the two greatly affects functions of music (e.g. narrative). Within cultures, many people have the same cultural embeddings which may result in similar compositional and listening grammars. For example, the transmission success of narrative in music greatly depends on this. However, the composer, no matter how hard s/he might try, cannot guarantee that an intended message—be it narrative or anything else—will be properly received. Experimental music often embraces this noise in the system (unknowable outcomes [21]), while narrative music often tries to minimize it. Pop music, for example, frequently references (often overtly) common cultural em-

beddings. Generally, similarities in compositional and listening grammars contribute to status-quo opinions about certain works within a culture and differences in tastes between cultures. The primary focus of this text is the objectivity (or subjectivity) and accuracy of various analytical methods and is not at all concerned with functions of music and musical structures.

1.5 Measures of Structure; Analysis and Empiricism

Each message in the music making process represents a morphology, an ordered list of elements ultimately expressible as a binary string. These elements can be anything: musical events and materials such as sounds and silences; parameters of musical events like pitch, duration, timbre; or even larger-scale gestalts and sequences.

How does one determine the amount of structure of a message? According to Gregory Chaitin, redundancy (non-randomness) indicates structure (as opposed to randomness or complexity). He defines (program-size) complexity as the minimum amount of information needed to specify an object (such as a morphology of elements). Imagine the following communication system: the transmission is a computer program that encodes a message representing an object, the channel is the computer, the output of the program is the object itself. If the smallest program needed to compute the output is smaller than the output itself, the object has structure (has redundancy). Chaitin connects this to Shannon entropy in "A Theory of Program-Size Complexity Formally Equivalent to Information Theory" [27]. As shown above, Chaitin also proves that program-size complexity (also known as Kolmogorov complexity after Andre Kol-

mogorov, who independently came up with similar results as Chaitin) is incomputable (for more on program-size complexity, see [95, 25, 69]). Furthermore, he shows that there exist objects that are maximally complex, axioms that are not self-evident. One example of such an object is the halting probability (first defined by Chaitin), which is the probability that a computer program generated by tosses of a fair coin will halt. Since no algorithm determines whether an arbitrary program halts, there is no way to know with complete accuracy the probability that a random program halts.

Unlike the halting probability, music is often highly structured (not maximally complex). Still, because of the halting problem, there are fundamental limits on how accurately and objectively we can compute the similarity of two arbitrary pieces of music. Chaitin's findings promote an approach common to music analysis, an empirical one.

1.6 Encoding Structure

It is a difficult task to determine an object's amount of structure and yet another to encode the structure itself. Formalizing structural metrics requires encoding both the interrelations and redundancies of the objects under investigation. An ideal structural metric compares the size of the elegant (or minimal-size) programs needed to compute the respective objects. If all redundancies are removed (as in an elegant program), then the number of bits alone represents the level of complexity. Next, the interrelationships between the elements are by definition encoded because the output of the program is the object itself. In his book *Meta Math!* [33], Chaitin himself suggests

this. He explains that calculating the mutual information of elegant programs that create music would produce viable metrics on style. Mutual information is a measure of the amount of information two objects share. (Mutual information was originally defined by Claude Shannon [93], but has been extended by Chaitin in his theory of program-size complexity.) Also, in Chaitin's earlier book *Algorithmic Information Theory* [29], he writes in a footnote that his work relating program-size complexity and biology (see [26, 28]) applies to music.

Since there is no way to compute whether or not a program is elegant, an empirical approach comes in handy even though such analysis is subjective and, as explained above, results in structural encodings with larger amounts of information than necessary. Still, one can define an arbitrary type of relationship, analyze the object under investigation looking for said relationship and then calculate redundancies of that relationship. In this case, it is important to define meaningful relationships that are not completely spurious.

1.7 A Survey of a Few Methods of Musical Analysis with Respect to Program-Size Complexity

Significant work has been done in analyzing relationships and redundancies within a morphology. Some of Polansky's metrics use matrix representations of relationships (see [82]). Robert Morris has produced similar techniques [78]. With respect to redundancy, the study of compression schemes is a well established field dating back to the inception of Shannon information theory. David Huffman extends Shannon's

work in "A Method for the Construction of Minimum-Redundancy Codes" [59] (see also [53, 86] for other compression techniques and [61, 70] for music specific compression techniques).

The work of David Cope, which he calls "musical intelligence" [38, 39] uses a model of analysis that can be compared to the ideal of finding minimal programs in algorithmic information theory. Much of Cope's work involves computer simulations of past musical styles. His interests lie in the ability of a machine to mimic styles of composers such as Beethoven and Bach. While his approach is rather convincing—he has generated numerous stylistically accurate pieces of older composers—he does not necessarily abide by Tenney's bases for a musical theory explained in the preface. For instance, his approach is style-specific. Also, it is unclear if his analytical methods actually provide information on the rules that govern a piece.

Cope's methods proceed similarly to Hidden Markov Models (see [24, 48]) by Markov analysis on a database of pieces of a given style. Since Cope uses rather large databases to recombine new music from extant music, the generated pieces are bound to be stylistically accurate. Such techniques do not really encode the rules behind a given set of pieces but rather produce results that share statistical properties to the input database. For Cope's purposes, this is completely satisfactory. Such a technique is essentially the polar opposite of finding minimal programs. With minimal programs, a piece is computed using the least amount of information, whereas Cope's technique is predicated on large amounts of information (in the database). Also, while Cope maintains that he is using an evolutionary method, he actually does not seem to have a fitness test. Instead, as mentioned above, he ensures stylistic similarity by virtue of

the fact that the database from which he draws is large and has consistency. He does not compare the output with the source. While he can produce stylistically accurate results, he cannot produce a piece that is measurable distant from the style of the pieces in the database since there is no metric. At best, he can use a database consisting of pieces of various styles.

However, a structural metric (including one based on Markov analysis that will be shown later) can be used as a fitness test for evolutionary methods of generating music. When an evolutionary algorithm implements a structural metric for a fitness test (as presented in the chapter on structural mutations), then it is possible to evolve a piece measurably distant to other pieces.

Ultimately, finding a minimal program that produces a set of pieces is the 'lowest-level' approach because a minimal program is essentially the genome of a given object. Cope himself suggests this in his book, *Hidden Structure: Music Analysis Using Computers* [40]. Recombination strictly on the phenotypical level does not model (creative) evolution. Not only do composers borrow and steal, they also innovate. They make small conceptual leaps that propagate and result in drastic shifts in musical paradigms and styles.

The field of music information retrieval provides other analytical techniques comparable to the analysis of minimal programs (in fact, there are entire repositories dedicated to the organization of the multitude of publications on the topic, see [1]). Many of these projects such as the "Music Genome Project" compare large sets of feature vectors that are attributed to songs in order to achieve metrics between various pieces. Often the features are a mixture of objective (such as instrumentation) and

subjective (such as style) criteria. As opposed to Cope's method, such comparisons can actually provide a value of distance between two works. On the other hand, as with Cope's analytical methods, comparisons are not at all made between the genes of various pieces of music (despite what the title, "Music Genome Project," suggests). A 'feature' is a result of an object's genome. While such music information retrieval techniques have proven very effective for practical purposes such as suggesting songs that someone might liked based on a previous record, they do not elucidate the intrinsic rules of a piece at their most fundamental level.

1.8 What is to Come

In the following chapter, it is shown that various techniques implementing graph and/or information theory provide several solutions of varying practicality to the problem of encoding structures for comparison. These structural encodings are then used for structural metrics which are later implemented for (rather artificial) structural evolutions. In conclusion, an examination of the properties of artificial structural evolutions provides a stepping stone for a computational model of evolution that actually mimics properties of 'real-world' evolution.

Chapter 2

Structure

2.1 Structure and its Analogs: Local and Global Morphological Constraints

First, we reiterate our distinction between structure and morphology. Morphology, which is an ordered list of elements, is a completely uncompressed representation of an object. Structure is the set of rules and relationships between individual elements and groups of elements that govern the morphology. Abstractly, we define structure as a set of local and global morphological constraints to which a morphology adheres. Local morphological constraints are equivalent to relationships defined by rules applied to or observations occurring between adjacent elements in a morphology [104]. Global morphological constraints are equivalent to statistical properties of the entire morphology that account for redundancies of elements and relationships. These constraints can be encoded in several ways such as in a computer program that generates the morphology even just an abstract graph. Such representations closely relate to each other but may

differ in the amount of compression of the morphology (that is, the size of the structural encoding). For now, we focus on structural encodings with graphs as there exist several cogent examples of music using similar ideas. Such examples are provided further in this chapter.

Morphological constraints can be represented by a connected graph $G = (V, E)$ where V is a set of vertices and E is a set of vertex pairs. The graph is directed (see Fig. 2.1) if the vertex pairs are ordered and undirected (see Fig. 2.2) if otherwise. The vertices of the graph represent elements such as musical parameters and events. Edges represent satisfactions of local morphological constraints. Vertex and/or edge labels (often called weightings) can represent global morphological constraints by values that express the probability or number of times an element or relationship occurs in a morphology. A well known example of a labeled graph is a graceful graph (shown in Fig. 2.3). In a graceful graph, edge labels are the difference between labels of connected vertices such that the list of edge labels goes from 1 to n inclusive (n being the size of the set of edges).

Labels that represent probability or number of occurrences are essential to structural analysis because they imply redundancy. However, labels can also represent other characteristics of an object. Vertices and edges may have multiple labels: one that implies redundancy and others that represent additional properties. For example, a label can be assigned a value representing a particular type of element or connection that can be used for ordering elements and connections between two objects. Another rather arbitrary musical example is that edge labels can represent the duration of a

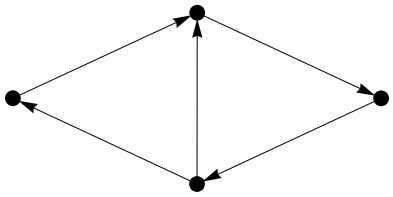


Figure 2.1: A directed graph

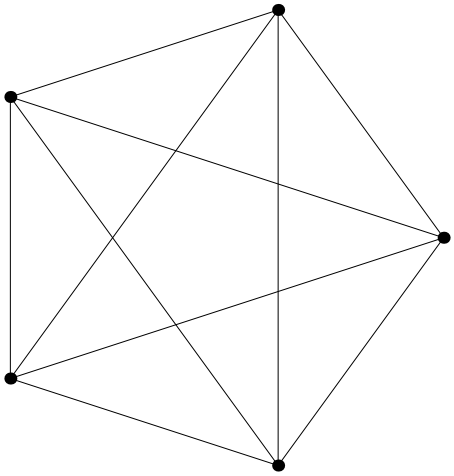


Figure 2.2: An undirected graph

musical event or the amount of time between two musical events. There are numerous such representations that may be useful for generative and analytical purposes.

A path through a graph represents a morphology. Note that local and global morphological constraints may conflict. A Hamiltonian cycle (see Fig. 2.4) is a morphology satisfying a well-defined global morphological constraint: the path visits each vertex only once and returns to the start vertex. However, not every graph contains a Hamiltonian cycle. Classifying graphs that contain certain types of paths such as Hamil-

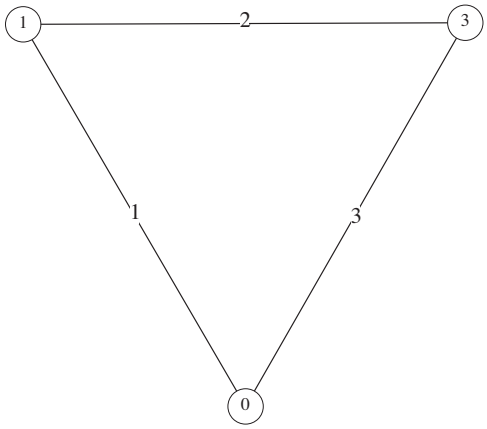


Figure 2.3: A labeled graph

tonian cycles is an important problem in graph theory (for examples, see [104, 41, 55]). In the case that labels represent probabilities, a long enough random walk on the graph will likely reflect the given distribution as a consequence of the law of large numbers [18].

2.2 Apriority and Posteriority

Before exploring examples of structural definitions in the creation of art, we note an important distinction between objective and subjective structural definitions with respect to concept and percept: an a priori structural definition applies to concept; alternatively, an a posteriori one applies to percept. A composer can define a piece based on a set of local and global morphological requirements that encodes an infinite set of possibilities; however, a realization of that piece is an instance, a "singular multiple" [14], of the original set. The conceiver defines a structure and derives a morphology,

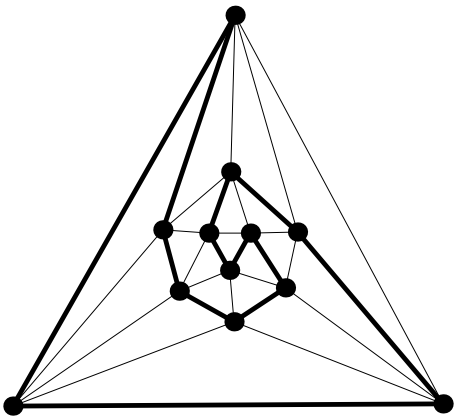


Figure 2.4: An icosehedron with a Hamiltonian cycle

whereas the perceiver is presented a morphology—the artifact itself—and must derive a structure (if they are so inclined).

We make this point because intuitively created art necessitates analysis of the more subjective, a posteriori definitions. Such definition are only fully validated when they can be used as a generator of the same results as a priori structural definitions or elegant programs. Such validation is ideal yet not necessary when taking an empirical approach.

Still, while a composer may define and encode structure for generative and analytical purposes, it is important to note that structure is real. It only exists in realization as the set of interrelations within the artifact itself. Mark So points out, "Composition, if it has to do with structure, doesn't exist before (in the score, in your head) but in the present of realization" [94].

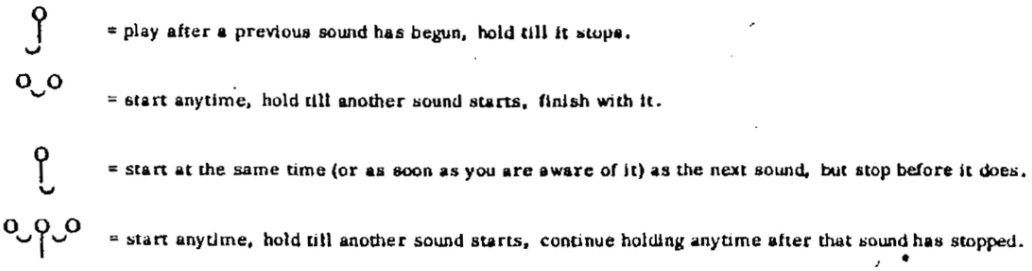


Figure 2.5: *For 1, 2 or 3 People* excerpt of instructions

2.3 Historical Premise

2.3.1 Inexplicit Examples

There exists a strong historical precedent for conceptual thinking about structure similar to the ways presented here. Composers have long been interested in structural design, computing music with machines and even using graph theory as explained above. The most abstract definition of structure—simply global and local morphological constraints—can be illustrated by pieces such as *For 1, 2 or 3 People* (1964) by Christian Wolff, *Psaltery* (1978–9) by Larry Polansky and *Arbor Vitae* (2006) by James Tenney. While there are many other examples, these were chosen for their clarity even though they do not adhere strictly to the notion of morphological constraints.

In *For 1, 2 or 3 People*, Christian Wolff defines musical events based on temporal relationships to other musical events. The notation instructs people to play before, during, or after another event. Essentially, the piece consists of a set of defined local morphological constraints that people spontaneously move through creating musical morphologies.

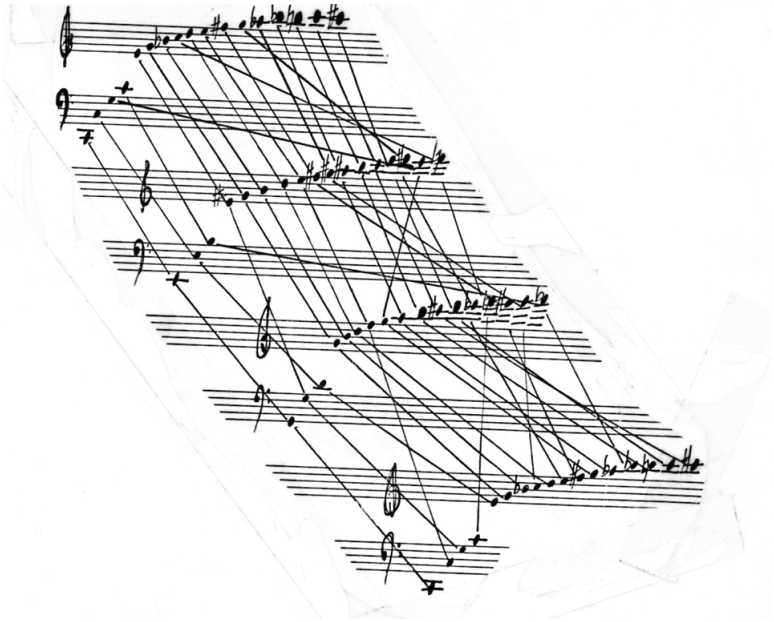


Figure 2.6: *Psaltery* sketch

In Polansky's piece *Psaltery*, harmonics of different fundamentals replace each other one by one. Polansky's original sketch for the piece (see Fig. 2.6) shows the replacement scheme by a graph where the vertices represent the pitches and the edges represent the replacements (which are similar to local morphological constraints).

When composing *Arbor Vitae* for string quartet, Tenney defined a harmonic lattice where the vertices represent harmonics of some fundamental and the edges connect harmonics related by one of a set of multiples—3, 5, 7 or 11. Tenney derived pitches from this lattice by stochastically choosing elements and then (sometimes) walking outward from the fundamental. A detailed description of this process and the statistical results are outlined in [103] and [83], respectively. Tenney first proposed such lattices in his essay "John Cage and the Theory of Harmony" [98]. While he does not mention

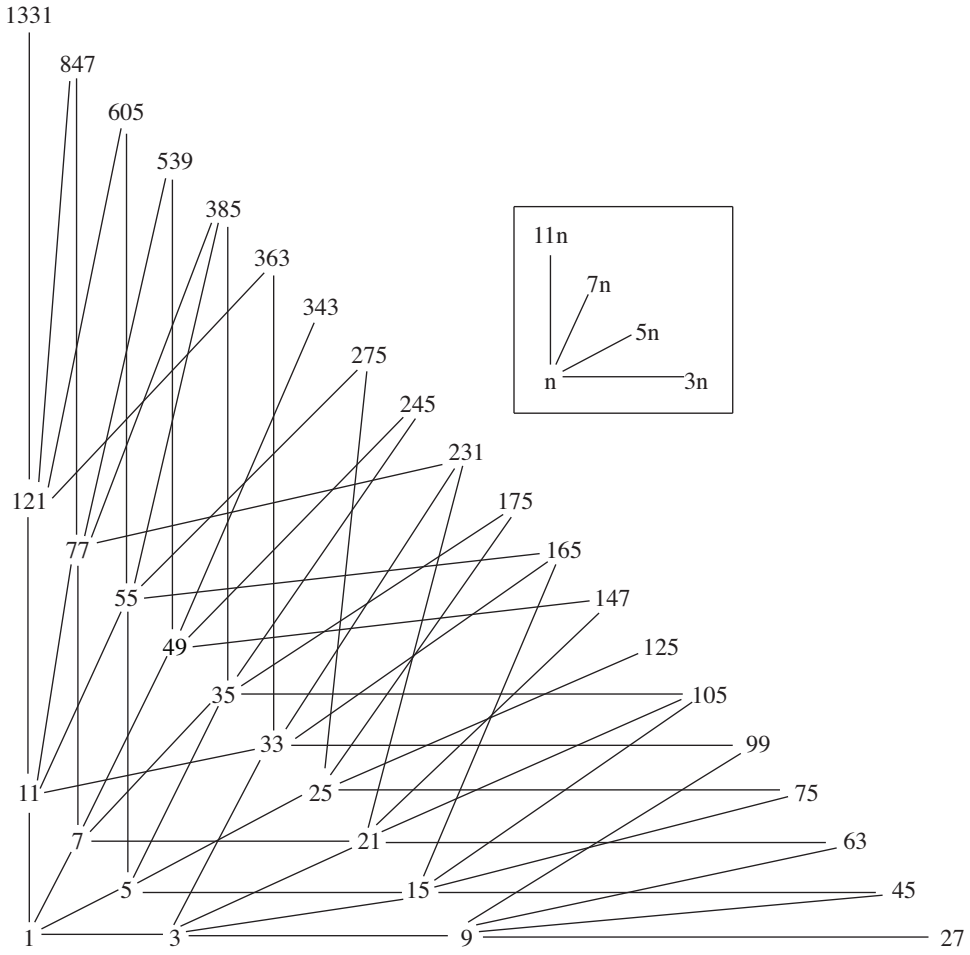


Figure 2.7: Harmonic lattice of *Arbor Vitae*

anything equivalent to global morphological constraints, the edge definitions (a set of multiples—typically frequency ratios) are essentially local morphological constraints. Tenney proposes how to "grow" such lattices in "On 'Crystal Growth' in Harmonic Space" [100]

2.3.2 Explicit Examples

Several artists explicitly make use of structures defined by local and global morphological constraints. Tom Johnson's piece, *Trio* (2005), is a prime example. The vertices/elements are three-note chords with pitches represented by numbers from 0 to 48, where middle C equals 24. The numbers of each chord are distinct partitions (without repetitions) of 72. That is, the constituent numerical representation of the pitches of each chord sum to 72. The edges are induced by the local morphological constraint that from chord to chord, one pitch must remain the same while the other pitches move by a semitone in contrary motion. The morphology is a Hamiltonian path; that is, satisfies the global morphological constraint that each vertex is visited only once. Fig. 2.8 shows the final morphology with the numeric representation of the chords. (Note that this graph does not show all the edges satisfying the local morphological constraint.) Johnson continues to consider similar ideas to generate music and has posed yet unsolved problems regarding the Hamiltonicity of other types of graphs in "Musical Questions for Mathematicians" [67].

Another example comes from the playwright Samuel Beckett. For his work *Quad* (1981), he wanted all combinations of performers on stage at some point such that the one that has been on stage the longest will always be the next to exit. This problem equates to finding a particular type of Gray code (after Frank Gray: an enumeration of all binary words of a given length such that only one bit flips from word to word, see [56, 68, 91]) where the bit position with the currently longest 'on' run will always be the next position to flip 'off.' Both local and global morphological constraints are

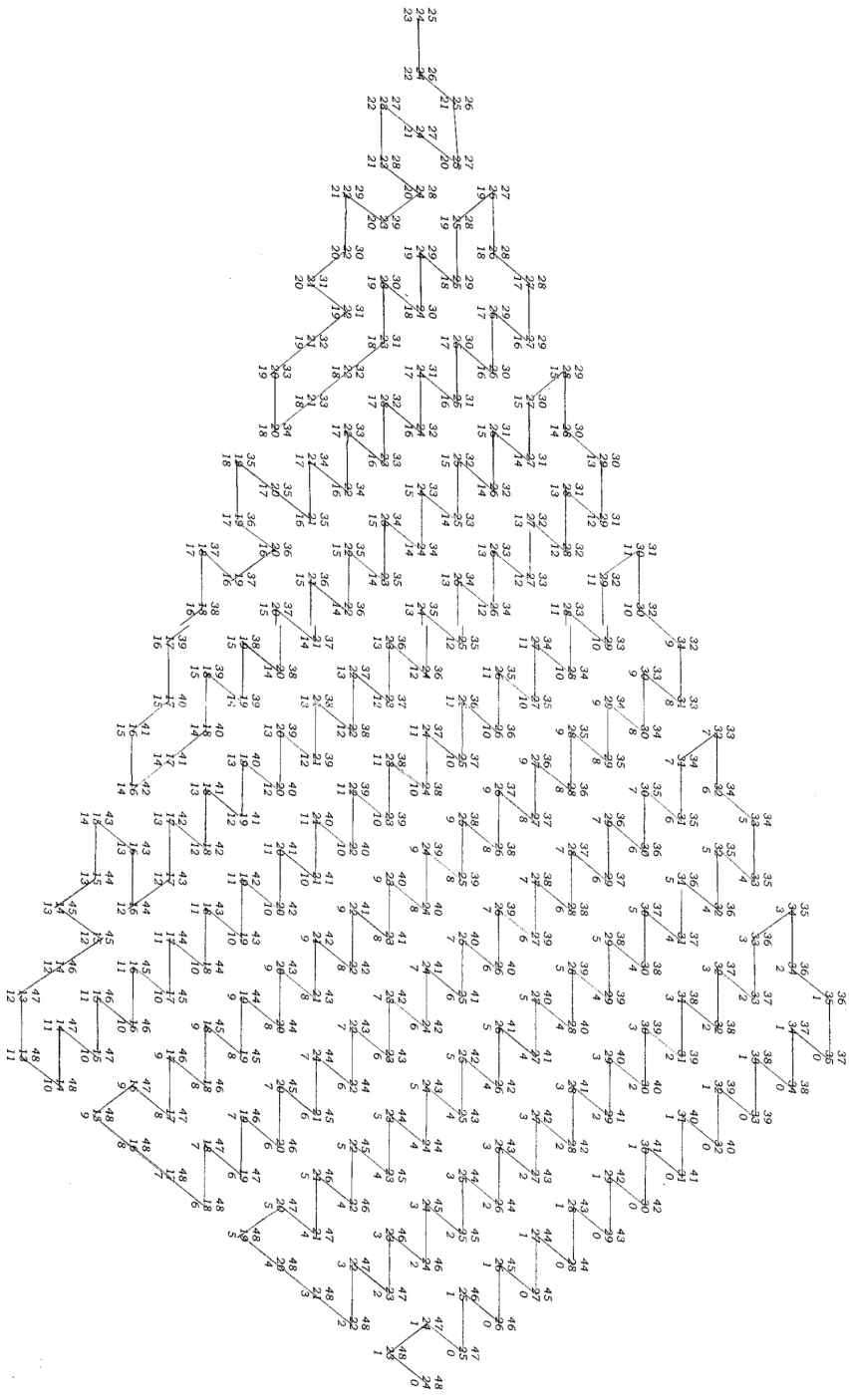


Figure 2.8: *Trio* graph (from cover of score)

0123456070121324356576071021353462670153741236256701731426206570
1342146560573102464537571020435376140736304642737035640271327505
4121027564150240365425013602541615604312576032572043157624321760
4520417516354767035647570625437242132624161523417514367143164314

Figure 2.9: The canonical transition sequence for an 8-bit Beckett-Gray code

well defined and easily expressible in mathematical terms. The local morphological constraint is that one bit flips between each two binary words. The global morphological constraints are that each binary word is visited only once and the current bit position with the longest 'on' run is the next to flip 'off.' For binary words of 3 and 4 bits, a Beckett-Gray code is impossible (a prime example of local and global morphological constraints conflicting) thus Beckett was unable to implement his original idea. Recently, Brett Stevens, et al. have shown that such Beckett-Gray codes exist for binary words up to 8 bits (excluding 3 and 4) and have provided an example of an 8-bit Beckett-Gray code [96]. Fig. 2.9 shows the aforementioned Beckett-Gray code's canonical transition sequence, which is an enumeration of the bit position in each word where the bit flip occurs.

2.4 A Few More Thoughts on Structure

Note that the definitions above are similar to graph and path representations of Markov processes and chains (see [92]), which are akin to finite state machines, which are essential to Turing machines. This illustrates that many structural encodings are valid and highly related. (For more on general graph, information theory, and algorithmic information theory, see [36], [93] and [29], respectively.) While each encoding is valid,

they all have limits when used to compare structures. Again, this text is essentially an epistemology of structural metrics.

Chapter 3

Metrics

3.1 Mathematical Definition

A metric is a means of determining similarity of objects using a distance function. Mathematically, it is a real-numbered function on a set S in the form $d(a, b)$ such that for any $a, b, c \in S$, the following conditions are satisfied:

1. $d(a, b) = 0$ if and only if $a = b$ (identity)
2. $d(a, b) = d(b, a)$ (symmetry)
3. $d(a, b) \geq 0$ (non-negativity)
4. $d(a, b) \leq d(a, c) + d(c, b)$ (triangle inequality)

Identity provides a standard of invariance. That is, the distance function equals 0 if the two elements being compared are evaluated as the same by the function. Symmetry and non-negativity state that order of comparison does not matter and the

result of the distance functions must always be positive, respectively. Take the natural numbers 1 and 3. If $d(a,b) = |a-b|$, then $d(1,3)$ and $d(3,1)$ both equal 2. Order does not matter and they are both positive. Triangle inequality states that the distance between a and b is at most the sum of the distance from a to c and c to b in a metric space, which is a set of objects being compared with a defined distance function. For our purposes, the objects will be representations of various structures. Note that the criteria for a metric can be relaxed. For example, a quasimetric satisfies all criteria except for symmetry. This can be useful in certain situations such as when the distance in one direction incurs an obstacle that is not present in the opposite direction (e.g. a quasimetric that calculates distance based on the amount of energy to get from one point to another will be different when going uphill as opposed to downhill). For more on metrics and metric spaces see [44, 97, 4].

3.2 Examples of Metrics

3.2.1 The Euclidean and Taxicab Metrics

The Euclidean and taxicab metrics are well known metrics where if $a = (a_1, a_2, \ldots, a_n)$ and $b = (b_1, b_2, \ldots, b_n)$ are two points in a Euclidean n-space, then the distance between a and b are respectively:

$$d(a,b) = \sqrt{(a_1-b_1)^2 + (a_2-b_2)^2 + \ldots + (a_n-b_n)^2} \qquad \text{(Euclidean)}$$

$$d(a,b) = |a_1-b_1| + |a_2-b_2| + \ldots + |a_n-b_n| \qquad \text{(taxicab)}$$

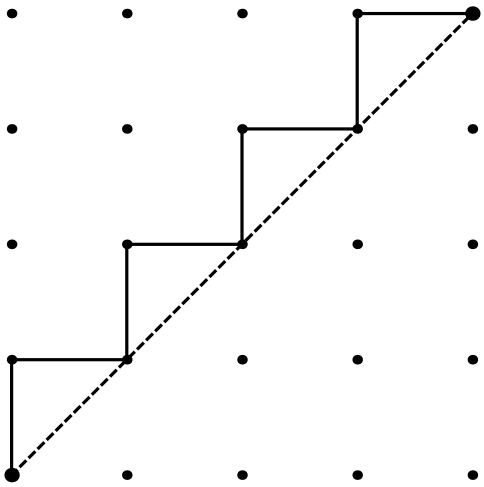


Figure 3.1: An illustration comparing the Euclidean and taxicab metrics

To show that metrics can vary considerably, note that relative to each other:

$$\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2} \leq |a_1 - b_1| + |a_2 - b_2| + \ldots + |a_n - b_n|$$

3.2.2 Harmonic Distance

Several composers and music theorists have been interested in determining harmonic distance between two frequencies. We review three examples of harmonic distance functions by Leonard Euler [49], Clarence Barlow [15] and James Tenney [98], respectively. The first, Euler's Gradus Suavitatis (GS) function, is a function determining integer complexity for a single value and harmonic distance when taken on the least common multiple of two values. The second harmonic distance function, defined by Barlow, is similar to Euler's GS in that first a calculation is made on a single value (called 'Indigestability') and then input into a distance function (called 'Harmonicity'). Note that Barlow's function is actually not a true metric since intervals that go up are

positive and down are negative. It can be turned into a proper metric by taking the absolute value of Harmonicity. The third metric, defined by Tenney, is perhaps the simplest. It takes the sum of the log values of the numerator and denominator of a given frequency ratio (in its simplified form). All three yield rather similar results and are essentially based on the complexity of the numbers in the frequencies ratio of two tones.

$$GS(a,b) = GS(ab)$$

$$\text{where } GS(n) = 1 + \sum_{i=1}^{k} v_i(p_i - 1)$$

for all prime factors of n such that $n = p_1^{v_1} p_2^{v_2} \dots p_k^{v_k}$ (Euler's GS)

$$H(a,b) = \frac{\text{sgn}(I(a) - I(b))}{I(a) + I(b)}$$

$$\text{where } I(n) = \sum_{i=1}^{k} v_i \frac{2(p_i - 1)^2}{p_i}$$

for all prime factors of n such that $n = p_1^{v_1} p_2^{v_2} \dots p_k^{v_k}$ (Barlow's Harmonicity)

$$HD(a,b) = \log \frac{a}{\gcd(a,b)} + \log \frac{b}{\gcd(a,b)}$$ (Tenney's Harmonic Distance)

3.2.3 Morphological Metrics

In "Morphological Metrics" [82], Polansky outlines ways to compare shapes and presents eight types of metrics that each have variants (see Table 3.1). His metrics compare "intervals" within each morphology being analyzed. An interval is a difference

	Magnitude Metrics		Direction Metrics	
	Linear	Combinatorial	Linear	Combinatorial
ordered	OLM	OCM	OLD	OCD
unordered	ULM	UCM	ULD	UCD

Table 3.1: The categories of Polansky's morphological metrics

between any two elements. The two broadest categories of Polansky's metrics are direction and magnitude metrics. The former compares general contour (up/down/sameness) between two morphologies and the latter, intervallic magnitude. Another distinction is linear and combinatorial. A linear morphological metric only compares intervals of adjacent elements within each morphology. A combinatorial morphological metric compares intervals between all elements within each morphology. The final distinction is ordered versus unordered which determines whether or not intervals between two morphologies correspond. Polansky's unordered, combinatorial metrics and multi-metrics (that combine linear and combinatorial metrics) actually approach structural metrics because they compare statistical properties of all intervals between two morphologies. Provided below are his general definitions of the four direction metrics for morphologies of the same length. $\Delta(x, y)$ determines interval and denotes any function measuring a difference between two elements $x, y \in M$ and $\rho(x, y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y \end{cases}$ is the discrete metric between any two values. See Polansky's paper for variants of these metrics, ways to account for morphologies of different lengths and definitions of the magnitude metrics not provided here.

$$ULD(M,N) =$$
$$\frac{1}{2(L-1)} \sum_{v=(-1,0,1)} \left| \sum_{i=1}^{L-1} L - 1 - \rho(\text{sgn}(\Delta(M_i, M_{i+1})), v) - \sum_{i=1}^{L-1} L - 1 - \rho(\text{sgn}(\Delta(N_i, N_{i+1})), v) \right|$$

$$UCD(M,N) =$$
$$\frac{1}{2(L-1)} \sum_{v=(-1,0,1)} \left| \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} L - 1 - \rho(\text{sgn}(\Delta(M_i, M_j)), v) - \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} L - 1 - \rho(\text{sgn}(\Delta(N_i, N_j)), v) \right|$$

$$OLD(M,N) = \frac{1}{L-1} \sum_{i=1}^{L-1} \rho(\text{sgn}(\Delta(M_i, M_{i+1})), \text{sgn}(\Delta(N_i, N_{i+1}))$$

$$OCD(M,N) = \frac{1}{\binom{L+1}{2}} \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \rho(\text{sgn}(\Delta(M_i, M_j)), \text{sgn}(\Delta(N_i, N_j))$$

3.2.4 Information-Based Metrics

The Hamming and Damerau-Levenshtein distances are two information-based metrics that warrant mention. The reader's knowledge of these will be useful for the structural metrics and structural mutations discussed later. The Hamming distance is

defined as the number of corresponding positions that differ between two strings of equal length [57]. For example, the Hamming distance between "**10**11101**0**" and "**01**11011**1**" is 3 (bold denotes the differing positions). The Damerau-Levenshtein distance is a metric between two strings defined by the number of operations needed to mutate from one string into another. The operations can be insertion, deletion, substitution and transposition of characters [42, 73]. For example, the Damerau-Levenshtein distance between "written" and "fitting" is 4:

1. written → ritten (delete 'w')
2. ritten → fitten (substitute 'f' for 'r')
3. fitten → fittin (substitute 'e' for 'i')
4. fittin → fitting (insert 'g' at the end)

Chapter 4

Structural Metrics

4.1 The Tradeoff: Objectivity for Practicality

In this chapter, we investigate three structural metrics ordered with respect to practicality of implementation starting with the least practical. The first structural metric we define is based on minimal program sizes. The second metric is based on the distribution of substrings in given morphologies and will be shown to be less discriminating than the program-size metric. The third metric compares graph representations of structures. Such an ordering reveals what seems to be a fundamental trade-off when analyzing structure; that is, objectivity versus practicality. More objective metrics are less practical to implement as will be shown clearly with the program-size based metric. There exists a spectrum of practicality that ranges from comparing the minimal programs that generate the objects under investigation to less compressed structural encodings where it may be difficult to determine the efficacy of compression with respect to maximal compression (such as graph-based structural metrics as opposed to

the minimal program-size based metric). Note that all metrics should be similar when comparing truly unstructured objects since all compression schemes will be equally ineffective. In these cases, the minimal program for a such an object simply prints each bit of the morphology one by one. While most arbitrary strings display this property, music, on the other hand, is often highly organized and thus finding structural encodings that approach minimal programs is ideal.

To reiterate from earlier in the text, this survey is for epistemological purposes only. What started out as a goal to provide objective, practical structural metrics and their applications turned into the goal of understanding why a truly objective, practical structural metric is essentially impossible. While some of the more practical structural metrics provided here may in fact prove very useful, implementations of the metrics have been omitted as they are not necessary to show the fundamental tradeoff of objectivity for practicality and to remain focussed on epistemology as opposed to utility.

As will be shown in the following chapter, a structural metric can be used as a fitness test for structural evolutions. Therefore, these studies extend to fields such as cognition and evolutionary biology and physics, where structural metrics are already being investigated [80, 52].

4.2 Mutual Information

One important idea for metrics briefly mentioned in the first chapter but excluded from the the previous chapter is mutual information. The mutual information of two objects is the amount of information they share in common, usually expressed in

bits. Shannon and Weaver first introduced this idea as a function of entropy [93], but calculating mutual information extends to other techniques as shown in the following sections.

Note that mutual information is actually not a metric. Similar objects have large amounts of mutual information. On the other hand, a metric expresses increasing similarity by smaller and smaller values. The following examples detail how one can use mutual information to calculate metrics.

4.3 The Most Objective Metric: Mutual Information of Minimal Programs

Chaitin applies the idea of mutual information to program-size complexity [27]. He defines mutual information in the theory of program-size complexity as the minimal amount of information needed to compute two objects separately minus the minimal amount of information needed to compute them together. If it takes about the same amount of information to compute the objects together as the sum of the information needed to compute them separately, then the two objects have a small amount of mutual information.

Chaitin defines the program-size complexity of a string or duple of strings as the minimal program needed to compute the string or duple of strings on a universal (self-delimiting) Turing machine started with a blank program tape expressed respectively as:

$$H(s) = \min|p|(U(p,\Lambda) = s) \qquad \text{(single string)}$$

$$H(s,t) = \min|p|(U(p,\Lambda) = (s,t)) \qquad \text{(duple of strings)}$$

The resulting definition of mutual information is:

$$I(s:t) = H(s) + H(t) - H(s,t) + O(1)$$

$O(1)$ denotes a function with an absolute value less than or equal to c for all values of its arguments. c essentially represents the number of bits for the set of simulators that enables a universal Turing machine to simulate all other (non-universal) Turing machines as an unspecified constant. Chaitin also shows that:

$$H(s) = -\log_2 P(s) + O(1)$$

$$H(s,t) = H(s) + H(t/s) + O(1)$$

$$H(t/s) = \log_2 \frac{P(s)}{P(s,t)} + O(1)$$

$P(s) = \sum 2^{-|p|}(U(p,\Lambda) = s)$ and $P(s,t) = \sum 2^{-|p|}(U(p,\Lambda) = (s,t))$ are the probabilities that a computer program generated by the toss of a fair coin will halt and output the string (or strings) on a universal (self-delimiting) Turing machine started with a blank program tape. $H(t/s)$ is the relative complexity of t given s expressed as the size of the minimal program to compute t if one is given the minimal program to compute s. Therefore:

$$H(s,t) = -\log_2 P(s,t) + O(1)$$

$$I(s:t) = \log_2 \frac{P(s,t)}{P(s)P(t)} + O(1)$$

As mentioned in the previous section, mutual information is not a metric but can easily be used for one. Simply take the joint complexity minus the mutual information:

$$\begin{aligned} D(s,t) &= H(s,t) - I(s:t) + O(1) \\ &= -\log_2 \frac{P(s)P(t)}{P(s,t)^2} + O(1) \end{aligned}$$

However, because computing whether or not random programs halt is impossible, $P(s)$ and $P(s,t)$ are incomputable; therefore, $H(s)$, $H(s,t)$, $I(s:t)$ and $D(s,t)$ are also incomputable. One can approach the upper bound of $D(s,t)$ by approaching lower bounds on $P(s)$ and $P(s,t)$, and indirectly, $H(s)$, $H(s,t)$, and $I(s:t)$. The process is rather simple and goes as follows. Given some time d, generate and run all programs of size less than or equal to d (there are 2^d many of these programs). After time d, check to see how many programs have halted and output s (or (s,t)). Any programs that have not halted or halted with a different output by time d are considered failures. The number of successes divided by 2^d approximates $P(s)$ (or $P(s,t)$). Chaitin has shown the same results in his book, *Exploring Randomness* [32]. He expresses the metric differently:

$$D(s,t) = H(s/t) + H(t/s) + O(1)$$

Chaitin uses relative complexities in his expression but this is equivalent to the definition above that uses probabilities:

$$\begin{aligned} D(s,t) &= H(s/t) + H(t/s) + O(1) \\ &= 2H(s,t) - H(s) - H(t) + O(1) \\ &= H(s) + H(t) - 2I(s:t) + O(1) \\ &= H(s,t) - I(s:t) + O(1) \\ &= -\log_2 \frac{P(s)P(t)}{P(s,t)^2} + O(1) \end{aligned}$$

Note that in the expression of $D(s,t)$ that uses probabilities, the calculation contains a divisor and thus (most likely) does not monotonically decrease, which results in local minima. That is, the metric does not get more and more accurate as d gets larger. Chaitin and I have discussed these results and he has proposed altering the metric above to produce a quasi-metric that is monotonic as d gets larger. This is achieved by using a variant definition of relative complexity where $H(s/t)$ is the minimal program to compute s if one is given t *directly* as opposed to the 'official' algorithmic information theory version where $H(s/t)$ is the size of the minimal program to compute s if one is given the *minimal program* to compute t.

Regardless, finding minimal programs is a huge software space to traverse and rather unfeasible. As computer speeds and memory increase, it is conceivable to let such a program run in the background calculating distance with great precision over time for a very usable structural metric.

4.4 Normality: A Metric that Accounts for the Distribution of Substrings.

A b-normal number is a real number with digits that are uniformly distributed in base b. Numbers are absolutely normal if the number is normal in every base. For the most part, normal numbers appear to be quite random. It seems that there are many computable numbers like π and e that are normal. While there have been no proofs that π and e are normal, Verónica Becher and Santiago Figueira show an example of a computable absolutely normal number in "An Example of a Computable Absolutely Normal Number" [17].

In music and many real-world objects, patterns often exist. Each substring does not occur with equal frequency. A structural metric that accounts for the distribution of substrings seems to be a good compromise for musical purposes. According to program-size complexity, digital expansions of numbers like π and e are very simple. A small program computes their respective digital expansions even though these numbers seem very random on the surface and are possibly absolutely normal. This is also the case with chaotic random number generators in most computer languages. The programs to generate them are rather small, but are so chaotic that it would take a very long time to notice any repetition. By the strict definition above, numbers are either normal or not. For our purposes, we should think of normality as a spectrum. Our 'normality tests' will determine where on this spectrum a sequence lies as opposed to whether or not a number is strictly normal. Strings with a distribution of substrings that approach the equal distribution of normal numbers seem more random. Such a

metric will still show similarity between two objects computed by minimal program of different sizes even if the object with the smaller minimal program is actually just chaotic.

The mathematical definition of an infinite normal string S^{∞} is:

$$\lim_{n\to\infty} \frac{N(S,n)}{n} = \frac{1}{b^k}$$

for all S where $N(S,n)$ is the number of times the finite string S in base b occurs in the first n digits of S^{∞}. For an absolutely normal number this is the case for all b.

Before moving on, it is necessary to spell out a few definitions in Shannon information theory. Shannon entropy for a string S is defined as:

$$H(S) = -\sum_{s\in S} p(s)\log_b p(s)$$

where $p(s)$ is the probability that the character s occurs in S. Mutual information in Shannon information theory is defined differently than in Chaitin's theory of program-size complexity:

$$I(S:T) = -\sum_{s\in S}\sum_{t\in T} p(s,t)\log_b \frac{p(s,t)}{p(s)p(t)}$$

where $p(s,t)$ is the probability that s occurs in S and t occurs in T at the same time.

The entropy of the sequence of digits for a b-normal number is 1, but that does not really tell us much. For example, the string "10101010101010" has an entropy of 1 but is obviously very structured. If one examines the next hierarchical level by considering that the string was output by a 1st order Markov source, then the string has 0 entropy. A much more important quality about a normal number is that the entropy rate on all hierarchical levels from 0 to ∞ (considering it as a Markov source

of order 1,2,...,∞) is also 1. At each hierarchical level, a normal number has maximal entropy. Shannon shows the entropy rate $H_n(S)$ of a Markov source of order n is as follows:

$$H_0(S) = -\sum_{i \in S} p(i) \log_b p(i)$$

$$H_1(S) = -\sum_{i \in S} p(i) \sum_{j \in S} p_i(j) \log_b p_i(j)$$

$$\vdots$$

$$H_n(S) = -\sum_{i \in S} p(i) \sum_{j \in S} p_i(j) \ldots \sum_{y \in S} p_{i,j,\ldots,y}(z) \log_b p_{i,j,\ldots,y}(z)$$

where $p_{i,j,\ldots,y}(z)$ is the probability that z occurs given the ordered set $\{i, j \ldots, y\} \in S$ as previous characters. For a normal number, $\sum_{n=0}^{\infty} H_n(S)$ diverges linearly as n approaches ∞. It is completely ergodic across all hierarchical levels.

Now we extend this principle to mutual information of two finite strings. Related methods have been produced by Dieter Arnold et al. [12], Andrea Goldsmith and Pravin Varaiya [51] as well as Tim Holliday et al. [58]. Two independent normal numbers should have no mutual information on all hierarchical levels. For now, we assume the strings have the same length. First, we derive the joint entropy rates, $H_n(S, T)$, and mutual information, $I_n(S : T)$, of hierarchical level n as follows:

$$H_0(S,T) = -\sum_{i\in S}\sum_{j\in T} p(i,j)\log_b p(i,j)$$

$$H_1(S,T) = -\sum_{i\in S}\sum_{j\in T} p(i,j)\sum_{k\in S}\sum_{l\in T} p_{(i,j)}(k,l)\log_b p_{(i,j)}(k,l)$$

$$\vdots$$

$$H_n(S,T) = -\sum_{i\in S}\sum_{j\in T} p(i,j)\sum_{k\in S}\sum_{l\in T} p_{(i,j)}(k,l)\ldots$$
$$\sum_{y\in S}\sum_{z\in T} p_{(i,j),(k,l),\ldots,(w,x)}(y,z)\log_b p_{(i,j),(k,l),\ldots,(w,x)}(y,z)$$

and

$$I_0(S:T) = -\sum_{i\in S}\sum_{j\in T} p(i,j)\log_b \frac{p(i,j)}{p(i)p(j)}$$

$$I_1(S:T) = -\sum_{i\in S}\sum_{j\in T} p(i,j)\sum_{k\in S}\sum_{l\in T} p_{(i,j)}(k,l)\log_b \frac{p_{(i,j)}(k,l)}{p_{(i,j)}(k)p_{(i,j)}(l)}$$

$$\vdots$$

$$I_n(S:T) = -\sum_{i\in S}\sum_{j\in T} p(i,j)\sum_{k\in S}\sum_{l\in T} p_{(i,j)}(k,l)\ldots$$
$$\sum_{y\in S}\sum_{z\in T} p_{(i,j),(k,l),\ldots,(w,x)}(y,z)\log_b \frac{p_{(i,j),(k,l),\ldots,(w,x)}(y,z)}{p_{(i,j),(k,l),\ldots,(w,x)}(y)p_{(i,j),(k,l),\ldots,(w,x)}(z)}$$

where $p_{(i,j),(k,l),\ldots,(w,x)}(y,z)$ is the probability that y occurs in S and z occurs at the same time in T given the ordered sets $\{i,k,\ldots,w\} \in S$ and $\{j,l,\ldots,x\} \in T$ as previous characters in each, respectively. n goes from 0 to the length of the strings.

A structural metric, $D_n(S,T)$, can be constructed for each hierarchical level:

$$D_n(S,T) = H_n(S,T) - I_n(S:T)$$

A more global structural metrics results from taking the average of several or all hierarchical levels up to the length of the strings:

$$D(S,T) = \frac{D_i(S,T) + D_j(S,T) + \ldots + D_n(S,T)}{l}$$

where $\{i, j, \ldots, n\}$ is some subset (of size l) of all hierarchical levels up to the length of the string. (Certainly there are methods other than the averaging one given above.) Also, the various hierarchical levels can be weighted such that:

$$D(S,T) = \frac{D_i(S,T)(w_i) + D_j(S,T)(w_j) + \ldots + D_n(S,T)(w_n)}{l}$$

where $\{w_i, w_j, \ldots, w_n\}$ is a vector of weights for the chosen hierarchical levels.

Now we must consider strings of different lengths. Note that the following proposed method also works on strings of the same length and may even provide a better solution. The method goes as follows. For each hierarchical level up to the length of the shorter string, perform a Markov analysis (of order equal to the hierarchical level) on the two strings. Then, generate Markov chains of the same length (the longer the chains, the better) from the respective transition probabilities. So if S_z^* and T_z^* are Markov chains constructed from the transition probabilities of S and T, respectively; as the chains grow larger, they should more accurately approach the marginal and joint distributions of the original strings. Mutual Information on any hierarchical level larger than the length of the shorter input string and less than or equal to the length of the longer input string will clearly be 0. For chosen hierarchical levels in this range, we just need to sum the respective entropy rates of the larger string (or Markov derived string) at those levels. If S is the longer string, then the metric is:

$$D(S,T) =$$
$$\frac{D_i(S_i^*, T_i^*) + D_j(S_j^*, T_j^*) + \ldots + D_n(S_n^*, T_n^*) + H_u(S_u^*) + H_v(S_v^*) + \ldots + H_z(S_z^*)}{l}$$

where $\{i, j, \ldots, n\}$ is some subset of hierarchical levels up to the length of T and $\{u, v, \ldots, z\}$ is some subset of hierarchical levels greater than the length of T and less than or equal to the length S. In this case, l is the size of the combined aforementioned sets. Also, weightings are still applicable.

This metric is quite objective. It will not show structural similarities to the depth of the metric based on program-size complexity, but it may better reflect how we perceive order as humans. Also, the length in time of the procedure, while still possibly very time consuming, is bounded by the length of the strings or the length we grow the derived Markov chains (granting that the longer we grow them, the more accurate the result such that eventually the increase in accuracy will be negligible unlike the program-size complexity model).

4.5 Graph Comparison

4.5.1 The Leap Into the Truly Subjective

The past two methods of structural comparison are quite objective. The first examines small computer programs that output given objects (which is essentially an investigation into maximal compression). The second investigates entropy-based calculations on given strings. Now we look at representations of structure using graphs. It was shown in the second chapter that graphs can suitably represent local and global

morphological constraints. One can name relationships, look for occurrences in the object of the named relationship and then derive an appropriate graph. This is the point where subjectivity enters the scene. Defining a relationship can be quite arbitrary. What is important (as mentioned earlier) is that the chosen relationships should avoid complete spuriousness. Any two elements can be deemed related by just saying so, but for musical (or most) purposes it seems logical that one deem two elements related if our observations suggest so. It is with this method that we clearly extend from the domain of the objective into the realm of subjectivity. We must start relying on what many musicians already rely on: intuition.

We must trust that those doing the analysis have defined relationships that many will find meaningful or cogent. Here we encounter the most crucial problem with several methods of musical analysis to date: there is no consensus on a set of meaningful relationships in music. The definition of a meaningful relationship is entirely subjective and differs from person to person. The methods in the previous sections avert this problem by a comprehensive approach. For example, the minimal program that generates an object is completely oblivious to its task but still performs the necessary calculations for a given computation. For methods based on graph theory, we must assume that a set of proper relationships have been determined. We must ultimately trust our (or others') intuitions in the initial step of the process—the derivation of the graph that represents a given object's structure.

4.5.2 Extensions

The following metrics based on graph theory extend to a wide variety of applications such as circuit and computer program optimization as well as large network analysis. For example, the World Wide Web is essentially a large network. Web pages and corresponding links are a structure where pages are structural elements and links are edges. These studies on scale-free and small-world networks have provided valuable results in the understanding of complex networks with applications that extend to situations such as mass transit and genomic biology (for examples, see [3]). Server-to-server propagation is another example where servers are structural elements and server-to-server connections are edges. Graph metrics have already been implemented to research computer program efficiency [79, 89, 90]. In these studies, graphs represent programs. The elements are the classes and edges are method calls from one class to another. Such metrics can be applied to boolean circuits as well. Note that if one knows the boolean circuit for a given musical output, the following methods may be implemented in a more objective fashion.

4.5.3 Graph Representations and Correlate Metrics

Since this is not a dissertation on innate perception and cognition of music, we must continue assuming the viability of the derived graphs that represent the structures of the objects under investigation. We will focus on three fundamental representations of graphs for calculating corresponding metrics: the degree set, the adjacency matrix and the distance matrix.

For a graph G with vertex set $V(G) = \{v_1, v_2, \ldots, v_n\}$ and edge set $E(G) = \{e_1, e_2, \ldots, e_m\}$, the degree set of a graph $D(G) = \{d_1, d_2, \ldots, d_n\}$ is the number of incident edges to each vertex $v_1, v_2, \ldots, v_n$.

Continuing, the adjacency matrix is an $n \times n$ matrix $A(G) = [a_{ij}]$, where:

$$a_{ij} = \begin{cases} 1 \text{ if } v_i v_j \in E(G) \\ 0 \text{ if } v_i v_j \notin E(G) \end{cases}$$

The distance matrix is highly related to the adjacency matrix. It does not simply show if two vertices are connected, rather it shows the 'cost' of getting from one vertex to another. It is is an $n \times n$ matrix $C(G) = [c_{ij}]$, where c_{ij} is the geodesic distance between the two vertices, which is the number of edges crossed on the shortest path between two vertices.

Again, note the similarity between our various representations and subsequent metrics: an adjacency matrix convolved with a set of probabilities is the same as the transition matrix for a Markov process. Also, a graph can represent a finite state machine which is at the heart of all Turing machines. There is no doubt that all the metrics in this chapter are highly related.

For a reasonable structural metric that uses any of the graph representations, the metrics should somehow show graph invariances and isomorphisms of subgraphs. Subgraph matching is a long studied problem in graph theory literature. It is known to be an np-complete problem, which is a type of problem that can be verified in polynomial time, but seems unlikely to be solvable in polynomial time (a proof has yet to be produced and there is an acclaimed prize in mathematics for one) [37]. We will not concern ourselves with the efficiency of such an algorithm. No matter what, matching

subgraphs can be found in some (possibly long) amount of time (unlike in program-size complexity where there is no algorithm to find a minimal program with utter certainty). A few metrics will be illustrated, but certainly many more can be derived. First, we discuss metrics on structures with the same amount of structural elements (or vertices).

The following metric can be used for any of the matrix representations:

$$D(G_1, G_2) = \min_{k=1}^{k!} \left(\frac{\sum_{i=1}^{n} \sum_{j=1}^{n} d(X_{ij}, (P_k^{-1}(YP_k))_{ij})}{s} \right)$$

where $d(X_{ij}, (P_k^{-1}(YP_k))_{ij})$ is a distance function on the corresponding indices of the matrices; for example, $\left| d(X_{ij} - (P_k^{-1}(YP_k))_{ij} \right|$. $\{P_1, P_2, \ldots, P_{n!}\}$ is the set of all $n \times n$ row-column permuted matrices, and s is some scaling factor (for comparing adjacency matrices, if $s = n^2$ then the metric should fall between 0 and 1). Permuting one of the matrices is necessary in order to find the most similar correspondence between vertices of the two graphs. Furthermore, the matrices can be weighted by multiplying each of them with a weightings matrix.

The degree set expresses a morphology of vertex connectivity. Technically, any of the metrics in the prior sections can be used for comparing degree sequences. I believe calculating the entropy of degree sequences is similar to what Tenney refers to as "structural entropy" at the end of *META Meta + Hodos*. At the time he wrote it, he pointed out that "nothing is known about structural entropies" [99]. Furthermore, any of the matrix representations can have the rows concatenated into a single string (or successive tuples of strings) and then be subject to the prior methods. Or, both distance and adjacency matrices (as well as ones that are multiplied with a set of weightings)

can be used as transition tables for Markov chains. Strings can be generated from these converted matrices and then subjected to the metrics in the prior sections.

To analyze graphs with different amounts of vertices, we must pad the smaller graph by adding a set of unconnected vertices. For the adjacency matrix, this simply means padding the smaller matrix with zeroes until it is the same size as the larger matrix. For the distance matrix, the smaller matrix must be padded with some high constant since a disconnected vertex technically is infinitely distant to all other vertices. Finally, for the degree set, the smaller set can be padded with zeroes until it is the same size of the larger set because a disconnected vertex has no incident edges.

Clearly, metrics greatly depend on how the objects are encoded. This will be discussed in even more detail later. For now, it should start to become clear that there is a fundamental difference of subjectively naming perceived relationships of an object as opposed to comprehensive analyzing a complete representations of the objects (such as finding the minimal programs for lists of objective characteristics like amplitude values of a sound file).

4.6 A Few More Well-Studied Ideas: Compression Schemes, Cross Correlation and Some Others

There are a few other techniques useful for structural metrics including compression schemes and cross-correlation. Our program-size complexity model strives for minimal programs, but certainly there are several (although perhaps less ideal) com-

pression schemes that are very practical and extensively in use. If $|C(M)|$ is the size of a morphology after submission through some compression scheme, then a metric (for any compression scheme) can be derived:

$$D(M,N) = |N| + |M| - |C(N)| + |C(M)| - |C(M,N)|$$

$|C(N)|+|C(M)|-|C(M,N)|$ is a type of mutual information on two morphologies given an arbitrary compression scheme and $|N| + |M|$ is the total length of the combined strings.

Cross-correlation is defined as the correlation of one morphology at the delay of another:

$$R(M,N) = \sum_{i=1}^{n} (M_i - \mu(M))(N_{i-d} - \mu(N))$$

where $\mu(M)$ is the mean of a given morphology of length n and d is some delay of elements. To turn this into a metric, one can sum the inverses of all correlations at delay d less than n. Correlating a morphology with itself has long been used in the Fourier analysis of sound signals to go from the time domain of a signal to the frequency domain. This perhaps suggests that cross-correlation of two signals (as opposed to autocorrelation) might provide a very suitable structural metric. (For more on Fourier analysis and cross-correlation see [19, 20, 87].)

As mentioned previously, some of Polansky's morphological metrics seem to approach structural metrics. Also, the metrics provided at the end of the third chapter (especially the Damerau-Levenshtein distance) might be very suitable structural metrics in certain cases.

4.7 Perhaps a Few Tricks for Analyzing Music

Until now, there has been little to no discussion on exactly what we are comparing apart from morphologies possibly expressed as bit strings. This is a very 'low-level' approach. For music, there are several ways to compare musical morphologies. What is most important is that the information is objective. For example, one can compare amplitude values of a sound file or lists of attributes of a piece such as frequencies and durations.

Furthermore, for each structural metric we need to understand the scope of analysis. Examining minimal programs is ideal because of its comprehensiveness. Lets take a simple example: a morphology of frequencies that has several repetitions of frequency ratios (or interval vectors) at different transpositions. A minimal program will account for that because it will most likely run a 'transpose' subroutine that takes any given sequence and shifts it up or down. Our normality test really does not have that capability when implemented with absolute values. The same sequence transposed is not identified as the same. However, it would be if the sequence was represented by relative values (the differences between adjacent elements of a sequence of absolute values). That way, the absolute values are subordinate to the relativities that make up the morphology.

Chapter 5

Structural Mutations

5.1 What is Being Mutated?

A structural mutation is the evolution of an object with a structural metric as a fitness test. Prior to the epistemological goals that the thesis now addresses, structural mutations were to be examined for their utility. Now, we study evolutionary methods (with structural metrics for fitness tests) to show how evolutionary algorithms as are usually implemented in computer science are far from similar to 'real-world' evolution (for more on general practices of evolutionary algorithms, see [13] and [47]).

Most importantly in the 'real-world' of evolutionary biology, mutations occur to the genome. The computational equivalent is a mutation of the machine-level code that generates a given object. In such a mutation, there is no way to ensure that the mutated code will halt. However, one can limit the software space by mutating the code on a higher level (i.e. not machine code) and embedding rules for mutation that take into account the grammar and syntax of a language to ensure that mutations will

not result in computations that loop infinitely and never halt. This essentially limits the software space and compromises the universality of the search. I maintain that this latter method, while perhaps useful, does not really explore epistemological limits since the software space is so greatly restricted. Again, the halting problem propagates itself in another situation.

In the following three examples, we examine artificial evolutions where the binary description/morphology is mutated for practicality (that is, on some higher level than the genome as with most evolutionary algorithms as we know them). Here we must distinguish between two very different utilities of practical evolutionary algorithms. The first utility is to induce a controlled morph from one object to the next (displayed by the first two algorithms below). The third algorithm relaxes control in order to reach the destination metric as quickly as possible. This is typical in problem-solving evolutionary algorithms where reaching the target metric equates to a solution of a given problem. Apart from mutations occurring on the phenotypical level, another reason why the first two of the following algorithms do not mimic 'real-world' evolution is because the trajectory of the evolution is so rigorously controlled. One fundamental principle of 'real-world' evolution is the occurrence of occasional big jumps in evolutionary progress. The reason why the third (problem-solving algorithm) does not mimic 'real-world' evolution is because once a solution is found, the algorithm finishes no matter if a better solution exists. Further optimization requires starting from scratch and losing the history of the last evolution. In the 'real-world,' organisms adapt to more effectively use less genetic information in order to find better solutions to problems that are already answered.

In the following chapter, we examine a new mathematical definition of evolution that maintains real evolutionary principles because of the halting problem. What has seemed to be a problem all along, the halting problem, may just be the very key that is needed to establish a proper mathematical definition of evolution.

5.2 Untargeted and Targeted Mutations with Controlled Trajectories

The primary distinction between types of mutations with controlled trajectories are untargeted and targeted. An untargeted mutation goes from a source morphology to an unknown morphology that has a specified distance (provided by the structural metric) from the source morphology. A targeted mutation goes from a source morphology to a target (or destination) morphology. While the two techniques are similar, the latter requires a few extra calculations.

5.3 The General Algorithm

5.3.1 Untargeted

1. Define a source morphology.
2. Define a structural metric $d(a, b)$.
3. Define a destination distance that the final mutated morphology must have from the source morphology.
4. Initialize the evolving morphology to be the same as the source morphology.

5. Define a function over time that goes from 0 to the destination distance such that each increment in time provides an ideal distance between the source morphology and the evolving morphology.

6. Mutate the evolving morphology several times.

7. Keep the mutation that best approximates the current ideal distance between the source and evolving morphology.

8. Repeat steps 6 and 7 until the final metric defined in step 3 is reached.

5.3.2 Targeted

The primary differences to an untargeted mutation are in steps 1, 3 and 7. Note that in a targeted mutation, structural metrics invariance must also result in structural isomorphism. That is, reaching the target metric must ensure reaching the target object.

1. Define a source and target morphology.

2. Define a structural metric $d(a, b)$.

3. Calculate the distance between the source and target morphology given the defined structural metric.

4. Initialize the evolving morphology to be the same as the source morphology.

5. Define a function over time that goes from 0 to the destination distance such that each increment in time gives an ideal distance between the source morphology and the evolving morphology.

6. Mutate the evolving morphology several times.

7. Keep the mutation that best approximates the current ideal distance between the evolving morphology and both the source and target morphologies.

8. Repeat steps 6 and 7 until the final metric defined in step 3 is reached.

5.4 Closer to Life: Problem Solving Algorithms

This method abandons overall control of the trajectory of a mutation in favor of reaching the target metric as quickly as possible. The following algorithm omits step 5 from the previous versions and alters the remaining steps:

1. Define a source morphology.

2. Define a structural metric $d(a, b)$.

3. Define a destination distance that the final mutated morphology must have from the source morphology.

4. Initialize the evolving morphology to be the same as the source morphology.

5. Mutate the evolving morphology several times.

6. Keep the mutation that makes the largest leap towards the destination distance.

7. Repeat steps 6 and 7 until the final metric defined in step 3 is reached.

5.5 A Stepping Stone to a Mathematical Model of Evolution

To reiterate, we presented the aforementioned algorithms to show how they do not model intrinsic properties of 'real-world' evolution. The first two algorithms do not incur large evolutionary advancements in order to control the trajectory of the mutation while the third algorithm does not optimize the solution once the target metric is reached. In all three, mutations occur on some higher level than the genome as to ensure that a mutation does not result in a non-halting program. That is, just as with the more practical structural metrics, these 'artificial' methods abandon fundamental principles of evolution for utility. In the next chapter, we abandon the notion of practicality to define a new mathematical model of evolution that maintains properties of 'real-world' evolution.

Chapter 6

Conclusion

6.1 Recasting a Mathematical Definition of Evolution

6.1.1 Examining Metabiology

In the previous chapter, mutations based on structural analysis were formulated in such a way that one can ensure that a mutation does not result in a non-halting program. But what about a computational model of evolution that is in keeping with observations in evolutionary biology and physics? There has been significant work in mathematical definitions of evolution for the past century. Chaitin himself started writing about such models as far back as 1970 and the theories of digital philosophy and physics also promote such computational models [26, 28].

Chaitin's most recent work returns to the intersection of mathematics and biology [34, 35]. He has dubbed the field "metabiology" and relates this work to the philosophical ideas of "the search for the perfect language" after Umberto Eco's book of the same title [46]. In short, the perfect language expresses concepts as simply

as possible. After examining Chaitin's current mathematical definition of evolution, I present a modified theory that, in no uncertain terms, is a search for the perfect language.

To start, we examine four integral features of 'real-world' evolution:

1. Organisms have a code such as DNA that contains a smaller amount of information than the organism itself. This code is the program that generates the object.

2. Systems have a tendency to grow more complex over time. For example, single-cell organisms develop into multicellular organisms however the increase in genetic material is rather small (also see below about genetic maintenance of history).

3. As something evolves, there tend to be occasional big jumps in evolution.

4. Phylogeny recapitulates ontogeny. While this general notion first outlined by Ernest Henkel (see [54]) is discredited, the development of something (like an organism) does seem to maintain aspects of its evolutionary history. Genomes become more efficient over time. Evolved species can do more with small amounts of DNA. For example, the human genome is much smaller than scientists originally expected [60]. The genomes of evolved species are not significantly larger than the genomes of primitive species (take humans in comparison to roundworms). Part of this efficiency seems to be a result of an encoding of an organism's evolutionary history. For example, humans have a proclivity to talk. A child can learn a language that has taken ages to develop in the matter of a few years. This suggests that these advancements are 'hard-coded' in the genome and not relearned from scratch thus expediting the process of learning language. The immediate result

is that genomes maintain certain aspects of the organisms history even if they do not obviously recapitulate their own evolution.

Chaitin's mathematical definition of evolution is beautifully simple. His model derives from Darwinian evolution of species. In the theory, the population comprises mutating software organisms that compute positive integers. The fitness test of an organism is the size of the integer it computes—the bigger the better [30]. More formally, Chaitin is searching a software space to find particular types of Turing machines that output large numbers (such machines are known as "busy beavers," a name applied by Tibor Radó [85]). There is no algorithm that will compute the output of such Turing machines and thus the problem of identifying such machines is equivalent to the halting problem .

Chaitin's model reflects the first three of four features above. The size of the integer that the software produces will grow much faster than the size of the computer program. Also, a change of a single bit in a computer program can cause a drastic change in output (big evolutionary jumps). As yet, Chaitin cannot show that his theory demonstrates the last of the four features in that the evolving code does not seem to contain its own history. He can show that evolution is progressing, but has to start the process over again and again if necessary (that is, the process is not cumulative). Furthermore, there is another obstacle. Non-halting programs are by definition maximally unfit. Here again, the halting problem has propagated itself. How does one avoid mutating to a program that does not halt? Chaitin's current theory suggests consulting an oracle that knows whether a given program will halt (which kind of suggests a di-

vine power). In the following section, I propose a theory in which a software organism maintains its history such that the halting problem is the reason why there are big evolutionary jumps.

6.1.2 Growing an Oracle: A New Mathematical Theory of Evolution

Again, Chaitin's model requires the consultation of an oracle that knows whether a program halts. In the following theory, a particular type of oracle is 'grown.' What do I mean by this? First, we return to the concept of the perfect language. Simply put, the perfect language is a language that can do everything with the smallest amount of information possible. More simply put, the perfect language is an oracle of all elegant programs. It is actually rather easy to grow such an oracle by extending Chaitin's program that approximates the probability that a random computer program will halt. This model, presented in the following points, is ultimately the culmination of my research in that it was completely engendered by the exploration of the epistemology of structural metrics.

- The fitness test: create an organisms that can do more with less. An organism's genotype is a list of halting programs. At first, the organism can do nothing as it is an empty list of programs.
- Search the software space to find programs that halt. (I believe the search could be a random walk or a systematic.)
- Let each mutant program run indefinitely.

- For each mutant program that halts, run all programs in the list (that is, in the organism's code).
- If the mutant program that halts is smaller than and produces the same output as a program already in the list, then replace the the existing program with the mutant one. (That way the organism's genotype gets leaner while maintaining its history but more efficiently over time.)
- If the new program produces a unique output, then just add it to the list. (This mimics the progression from 'single-celled' organisms to multi-cellular complexes.)

In this model, as time goes on, both the genome and the phoneme will grow but the former will grow at a much slower rate. Complexity will always tend to increase. Over time, the organism will maintain its history and get more efficient because it will slowly replace bigger programs with smaller, more elegant ones that do the same thing (that is, the process is cumulative). Major jumps in evolution occur when a new program outputs something unique with respect to the list and is then added to the organism's genome. Smaller evolutionary progressions will occur when the software organism becomes leaner as it gets rid of extraneous code. Essentially, the organism evolves towards a list (or oracle) of all minimal program. The evolution is the search for the perfect language.

6.2 Important Findings

Our epistemological survey results in two rather important findings.

- In structural analysis, a fundamental tradeoff of objectivity for practicality exists. The most objective metric is based on the minimal programs of given objects. A minimal program is maximally compressed. Due to the halting problem, knowing the minimal program of a given object is essentially impossible. That is, what we need to know to objectively measure differences in structure, we cannot know. All (other) methods of structural analysis that do not use maximally compressed representations of the objects under examination sacrifice the objectivity of the analysis. What is perhaps often gained is a practically efficient method.

- A mathematical theory of evolution should take into account the halting problem as the reason for evolutionary advancement in both minor and major evolutionary leaps.

These two ideas were completely unknown to me prior to starting. As I pointed out in the introduction, I began by trying to define objective structural metrics for practical reasons. As I got more lost in the research, I started to draw more lucid conclusions about what is possible. I make this point for no other reason than to show that sometime in order to find what we are looking for, we have to get completely lost. The notion of the experiment in art is substantially different from the traditional scientific process. Artists have nothing to prove. We need not make conjectures prior to embarking on an experiment. This process, while perhaps a bit haphazard, can lead us into the void and help us uncover truths and beauties that are not yet known. Both the experimental process in art and science are valid. Perhaps artists and scientists have a bit more to learn from each other than meets the eye.

6.3 Future Work and An Open Question

Further work is warranted in the practical application of structural metrics and mutations. I decided to omit practical examples as they were not needed to draw our epistemological conclusions and would have unnecessarily lengthened the text.

Included at the end of this text is an appendix that examines works that have informed and been informed by the research. Almost all of the pieces that implement graphs in their generation use rather small graphs with few vertices and edges. I have recently been extending these composition techniques to sonify complex networks much larger than anything I have tried so far. I hope to continue investigating new ways of using large-scale networks for generating music.

An open question that came about through my research is whether or not there exists a proper computational model of art. It is clear that for analytical purposes, knowing minimal programs that output given objects is ideal. Still, as mentioned in the beginning of *Structural Metrics*, art might be best characterized as those programs that are not efficient or minimal; programs large in size that may run for long amounts of time (disproportionate to their size) and output less information than is initially input into the computer. That is, what might characterize art is that each bit of information of the input actually tells us very little about the output. Of course, this is just one of many conceivable conjectures.

Curtis Roads suggests that aesthetic discrimination is intrinsic to our ability to survive and that art is a strategy for tuning our aesthetic discrimination [88]. Essentially, I agree, but want to further elaborate on the idea. Clearly, as art is ubiquitous in

human culture, it must exist for a reason. On the other hand, our need for aesthetic discrimination in order to survive is based on a direct correlation/consequence between the perceived stimuli and how it might affect us. For example, we need to be able to distinguish if something tastes foul in order to know that if we eat it, we will get sick. However, art tunes our aesthetic discrimination in a myriad of (possibly unknown) ways. I believe finding beauty in anything helps tune our aesthetic discrimination and, perhaps, this tuning happens just as effectively (if not more) when we are able to find beauty in the most unlikely of artifacts. Why else would I love Duchamp's work or that of the Viennese Actionist's and still be able to avoid pouring what smells like foul milk into my bowl of cereal? Art can be upsetting, disturbing, bland, boring, or any number of (typically negative) qualities yet still be appreciated, loved and admired as unconventionally beautiful (or perhaps as having a deep beauty as opposed to just a superficial prettiness). I distinguish very carefully between beauty and prettiness. The latter is sheer vanity and can be vacant while possibly pleasing. For example, a pretty person does not necessarily make a beautiful one and the other way around. So is the same with art.

Still, if art, in an information theory sense, is best characterized by large programs with long run-times and small outputs, then how is the information of a piece of art related to our need for it? Christian Calude has shown that such programs are quite rare in his paper, "Most Programs Stop Quickly or Never Halt" [23]. While art is ubiquitous, it is still rare. I believe that an interesting next step of research would be a proper examination of such programs and their epistemological consequence on artistic practices. Further, perhaps the necessity of art, especially if characterized

by large programs with long run-times and small outputs, is our ability to discern an innovative act. In conversation with my friend, composer Laura Steenberge, she made a very poignant connection regarding this information/program-size based view of art. In the oracle growing algorithm above, innovation can be characterized as a computer program that halts on an output that has yet occurred. At that moment, there may be no understanding of how the program works and if so, we can only appreciate it as art—for its beauty and innovation—as we may not know what the information of the program is telling us about regarding anything practical. Over time, we may find smaller and smaller programs that generate that same output, which might lead to a better understanding of the output and a resultant loss of novelty. Again, these are just conjectures made with the hope to stimulate more thought on these rather esoteric topics.

6.4 A Return to Philosophies on Music: Computation and Composition

> "It doesn't really matter if the viewer understands the concepts of the artist by seeing the art. Once it is out of his hand the artist has no control over the way a viewer will perceive the work. Different people will understand the same thing in a different way."
>
> Sol LeWitt from "Paragraphs on Conceptual Art" [74]

6.4.1 Ephemera

While the artist may employ intense logical rigor in conceptions of works, perceivers may and need not understand the concepts. Music, and art in general, is transcendent in that way. In the end, the concepts and ideas that may seem so integral

to the artist are just as ephemeral (with respect to the scope of the work) as any of the steps in the art making process. LeWitt reifies this by pointing out that "some ideas are logical in conception and illogical perceptually" [74]. This results directly from the incalculability of concept-to-percept transparency first introduced in the preface

Still, artists use logic in the end to make art. In "Sentences on Conceptual Art" [75], LeWitt states that "all ideas are art if they are concerned with art and fall within the conventions of art." In this text, mathematical objects are used to represent structures in art. For a while, math and art are in communication with each other. This communication is also ephemeral.

So why make art that is informed by other fields at all? In my mind making art is learning. This learning process is still a means to an end, but the end has little to no functional purpose other than to introduce into the world an artifact that has not yet been experienced—no personal gain, just a search for truth and beauty. I suppose truth is revealed in understanding whether or not something *can or cannot* exist and beauty is understood by experience through something that *does or does not* exist. Much of this text is based on what is computable, especially with respect to art. We assume that one important attribute of a successful computation is that it halts. If a machine halts, it is by definition ephemeral. It is not a perpetual machine, but rather a machine that flits into existence.

6.4.2 Scalability and Relativity

One can define relationships without defining absolute materials. This means that the concept is scalable to the resources available. Scalability is common in music. Composers have always explored variable ensembles, durations and other parameters.

Defining relationships relatively as opposed to absolutely is also prevalent in other art forms. One reason why this chapter has multiple references to Sol LeWitt is that much of his art is scalable. LeWitt often indicates a location for each element relative to the locations of other elements. In many of his "wall drawings," he provides a score realized and installed by a set of draftsmen. In these pieces, structural isomorphism from installation to installation can never be broken. The piece retains its structure independent of the size of the wall because it is defined relative to the size of the wall. It is completely scalable.

Such scalability results from relativities that are proportional as opposed to absolute. For example, telling a person to move three feet from their current location is not scalable; however, telling them to move a distance half the height of the closest person is. A scalable relation specifically in Sol LeWitt's work is provided in Fig. 6.2. In music, time dilation is a common example of scalability. Composers can prescribe scalable temporal relations between elements instead of assigning each element an absolute time.

When put in the context of computability, the concept of scalability is easily understood. A computer program can have a set of variables that, when changed, alter the piece, but do not change the structure of the piece. In concrete terms, modifying a

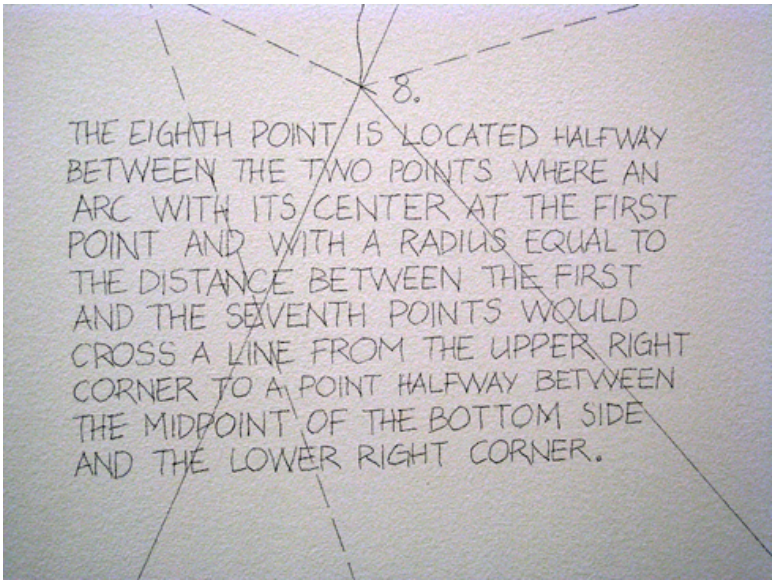


Figure 6.1: An instruction from Sol LeWitt's *Wall Drawing 305* (1977)

variable may have little affect on the size of the program, which can imply that there is a negligible change in the piece's structure. We can illustrate this idea through a piece of music that is scalable in length such as Tenney's *HAVING NEVER WRITTEN A NOTE FOR PERCUSSION* (1971). Changing the length of the piece (so long as it satisfies the instruction "very long") does not change the structure of the piece at all. If one considers the score as a computer program, changing the length only increases the size of the program logarithmically.

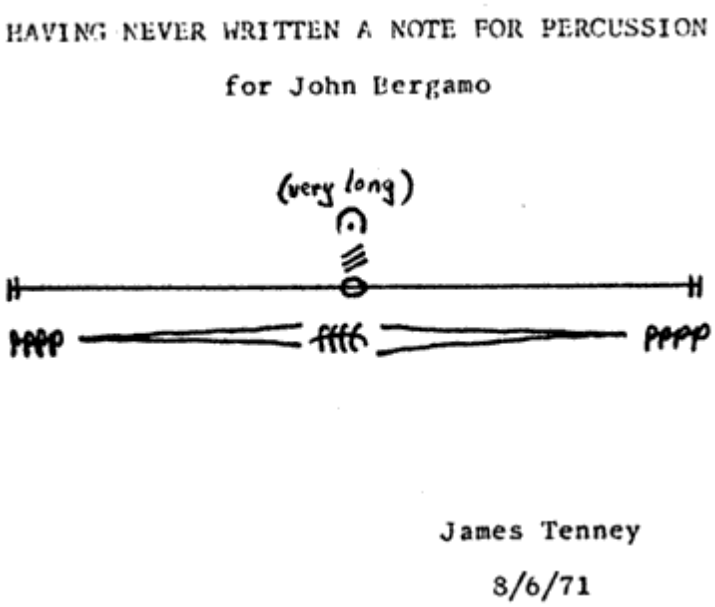


Figure 6.2: Score of James Tenney's *HAVING NEVER WRITTEN A NOTE FOR PERCUSSION* (1971)

6.4.3 On Documentation

Artists often obsess over documentation, especially musicians. For example, pieces are constantly recorded for safe keeping. One of the most important documentations in music is the score, which functions as a prompt to action and thus somehow encodes the possibilities of a piece. For most of western music history, the score has developed to best express said possibilities. Most recently, there has been a dramatic paradigm shift in the score with the proliferation of the computer and the incorporation of text. In many ways, it seems that the score tends towards a minimal or elegant description of a piece.

6.4.4 A Final Thought

Artists should be primarily concerned with art itself even if it is derived from other sources. If art becomes completely predicated on anything but art, the predicate risks becoming a virus that kills the host. I am happy to admit that I am first and foremost a composer who has been seduced into computer science and mathematics. Nonetheless, the simplest and most elegant conceptual ideas drawn from any source and applied to music have potential to yield interesting and seemingly complex results. Perhaps not coincidentally, this is often the case with mathematics.

Bibliography

[1] music-ir.org. <http://www.music-ir.org/>. accessed May 4, 2010.

[2] squaring.net. <http://www.squaring.net/>. accessed May 4, 2010.

[3] Reka Albert, Hawoong Jeong, and Albert-László Barabási. The diameter of the world wide web. *Nature*, 401:130–131, 1999.

[4] Luigi Ambrosio and Paolo Tilli. *Topics on Analysis in Metric Spaces*. Oxford University Press, Oxford, England, 2004.

[5] Charles Ames. Automated composition in retrospect. *Leonardo Music Journal*, 20:169–185, 1987.

[6] Charles Ames. Tutorial on automated composition. In *Proceedings of the ICMC*, pages 1–8. International Computer Music Association. Urbana, Illinois, 1987.

[7] Charles Ames. A catalog of statistical distributions: Techniques for transforming random, determinate and chaotic sequences. *Leonardo Music Journal*, 2:55–70, 1991.

[8] Charles Ames. Statistics and compositional balance. *Perspectives of New Music*, 28:80–111, 1991.

[9] Charles Ames. A catalog of sequence generators. *Leonardo Music Journal*, 2:55–72, 1992.

[10] Charles Ames. Thresholds of confidence: An analysis of statistical methods for composition, part 1: Theory. *Leonardo Music Journal*, 5:33–38, 1995.

[11] Charles Ames. Thresholds of confidence: An analysis of statistical methods for composition, part 2: Applications. *Leonardo Music Journal*, 6, 1996.

[12] Dieter Arnold, Hans-andrea Loeliger, and Pascal O. Vontobel. Computation of information rates from finite-state source/channel models. In *Proc. 40th Annual Allerton Conference on Communication, Control, and Computing*, pages 457–466, 2002.

[13] Thomas Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, Oxford, UK, 1996.

[14] Alain Badiou. *Being and Event*. Continuum International Publishing Group, New York City, New York, 2006.

[15] Clarence Barlow. *Musiquantics*. Feedback Press, Cologne, Germany, 2009.

[16] G. Douglas Barrett. Listening to language: the text scores of experimental music. preprint, 2010.

[17] Verónica Becher and Santiago Figueira. An example of a computable absolutely normal number. *Theoretical Computer Science*, 270(1–2):947–958, 2002.

[18] Jakob Bernoulli. *Doctrinae Civilibus, Moralibus & Oeconomicis*, chapter Ars Conjectandi: Usum & Applicationem. 1713.

[19] Ronald Bracewell. *The Fourier Transform and Its Applications*. McGraw-Hill Science/Engineering/Math, 3 edition, June 1999.

[20] E. Oran Brigham. *The Fast Fourier Transform and its Applications*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1988.

[21] John Cage. *Silence: Lectures and Writings by John Cage*. Wesleyan University Press, Middletown, Connecticut, 1961.

[22] Cristian Calude, Michael Dinneen, and Chi kou Shu. Computing a glimpse of randomness. *Experimental Mathematics*, 11:2002, 2002.

[23] Cristian Calude and Michael Stay. Most programs stop quickly or never halt. *Advances in Applied Mathematics*, 40(3):295 – 308, 2008.

[24] Olivier Cappé, Eric Moulines, and Tobias Ryden. *Inference in Hidden Markov Models (Springer Series in Statistics)*. Springer-Verlag New York, Inc., Secaucus, New Jersey, 2005.

[25] Gregory Chaitin. On the length of programs for computing finite binary sequences. *J. ACM*, 13(4):547–569, October 1966.

[26] Gregory Chaitin. To a mathematical definition of life. *ACM SICACT News*, 4:12–18, 1970.

[27] Gregory Chaitin. A theory of program size formally identical to information theory. *J. ACM*, 22(3):329–340, July 1975.

[28] Gregory Chaitin. *The Maximum Entropy Formalization*, chapter Toward a Mathematical Definition of Life, pages 477–498. MIT press, Cambridge, Massachusetts, 1979.

[29] Gregory Chaitin. *Algorithmic Information Theory*. Cambridge University Press, 1987.

[30] Gregory Chaitin. *Open Problems in Communication and Computation*, chapter Computing the Busy Beaver Function, pages 108–112. Springer-Verlag, 1987.

[31] Gregory Chaitin. The limits of mathematics—the book (in c). <http://arxiv.org/abs/chao-dyn/9407003v1>, 1994. accessed May 4, 2010.

[32] Gregory Chaitin. *Exploring randomness*. Springer-Verlag New York, Inc., Secaucus, New Jersey, 2001.

[33] Gregory Chaitin. *Meta Math!: The Quest for Omega*. Vintage, November 2004.

[34] Gregory Chaitin. Algorithmic information as a fundamental concept in physics, mathematics and biology. <http://www.cs.auckland.ac.nz/~chaitin/iqc.html>, 2009. accessed May 4, 2010.

[35] Gregory Chaitin. Mathematics, biology and metabiology. <http://www.cs.auckland.ac.nz/~chaitin/jack.html>, 2009. accessed May 4, 2010.

[36] Gary Chartrand and Linda Lesniak. *Graphs & Digraphs*. Chapman & Hall, New York City, New York, 2005.

[37] Stephen Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, New York, New York, 1971. ACM.

[38] David Cope. *Experiments in Musical Intelligence*. A-R Editions, Madison, Wisconsin, 1996.

[39] David Cope. *Computer Models of Musical Creativity*. MIT Press, Cambridge, Massachusetts, 2005.

[40] David Cope. *Hidden Structure: Music Analysis Using Computers*. A-R Editions, Madison, WI, 2009.

[41] Bill Cuckler and Jeff Kahn. Hamiltonian cycles in dirac graphs. *Combinatorica*, 29(3):299–326, May 2009.

[42] Fred Damerau. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176, 1964.

[43] Nicolaas Govert de Bruijn. A combinatorial problem. *Koninklijke Nederlandse Academie van Wetenschappen*, 49:758–764, 1946.

[44] Michel-Marie Deza and Elena Deza. *Dictionary of Distances*. Elsevier Science, Amsterdam, The Netherlands, October 2006.

[45] Adrianus Johannes Wilhelmus Duijvestijn. Simple perfect squared square of lowest order. *Journal of Combinatorial Theory, Series B*, 25(2):240 – 243, 1978.

[46] Umberto Eco. *The Search for the Perfect Language (The Making of Europe)*. Wiley-Blackwell, April 1997.

[47] Agoston Eiben and Jim Smith. *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.

[48] Yariv Ephraim and Neri Merhav. Hidden Markov processes. *Information Theory, IEEE Transactions on*, 48(6):1518–1569, Jun 2002.

[49] Leonard Euler. Tentamen novae theoriae musicae. *Typographia Academiae Scientiarum*, 1739.

[50] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik*, 38(1):173–198, 1931.

[51] Andrea Goldsmith and Pravin Varaiya. Capacity, mutual information, and coding for finite-state Markov channels. *Information Theory, IEEE Transactions on*, 42(3):868–886, May 1996.

[52] Robert Goldstone, Douglas Medin, and Jamin Halberstadt. Similarity in context. *Memory & Cognition*, 25(2):237–255, 1997.

[53] Soloman Golomb. Run-length encodings. *IEEE Transactions on Information Theory*, 12(3):399–401, 1976.

[54] Stephen Gould. *Ontogeny and Phylogeny*. Harvard University Press, Cambridge, Massachusetts, 1977.

[55] Sylvain Gravier. Hamiltonicity of the cross product of two hamiltonian graphs. *Discrete Math.*, 170(1-3):253–257, 1997.

[56] Frank Gray. Pulse code communication. U.S. Patent 2,632,058, filed November 1947.

[57] Richard Hamming. Error detecting and error correcting codes. *Bell System Techincal Journal*, 29:147–160, 1950.

[58] Tim Holliday, Andrea Goldsmith, and Peter Glynn. Capacity of finite state Markov channels with general inputs. In *Proceedings of the 2003 IEEE International Symposium on Information Theory*, page 289, 2003.

[59] David Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, September 1952.

[60] International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. *Nature*, 431(7011):931–945, October 2004.

[61] Nikil Jayant, James Johnston, and Robert Safranek. Signal compression based on models of human perception. *Proceedings of the IEEE*, 81(10):1385–1422, October 1993.

[62] Tom Johnson. Automatic music.
<http://www.editions75.com/Articles/Automatic%20music.pdf>, 1998.
accessed May 4, 2010.

[63] Tom Johnson. Self replicating loops.
<http://www.editions75.com/Articles/selfreplicatingloops.html>, 1999.
accessed May 4, 2010.

[64] Tom Johnson. Found mathematical objects.
<http://www.editions75.com/Articles/
Found%20Mathematical%20Objects.pdf>, 2001. accessed May 4, 2010.

[65] Tom Johnson. Tiling the line in theory and in practice.
<http://www.editions75.com/Articles/
Tiling%20the%20line%20in%20theory.pdf>, 2002. accessed May 4, 2010.

[66] Tom Johnson. Music and combinations.
<http://www.editions75.com/Articles/Music%20and%20Combinations.pdf>,
2003. accessed May 4, 2010.

[67] Tom Johnson. Musical questions for mathematicians.
<http://www.editions75.com/
Articles/Musical%20Questions%20for%20Mathematicians.pdf>, 2004. accessed
May 4, 2010.

[68] Donald Knuth. *The Art of Computer Programming.* Addison Wesley, 1997.

[69] Andre Kolmogorov. Three approaches to the quantitative definition of information. *International Journal of Computer Mathematics*, 2(1):157–168, 1968.

[70] Michael Krasner. Digital encoding of speech and audio signals based on the perceptual requirements of the auditory system. *MIT Lincoln Laboratory Technical*, 535, 1993.

[71] Joseph Kudirka. Composer/performer relationships in experimental music in regards to methods of notation. Master's thesis, University of Huddersfield, 2010.

[72] Fred Lerdahl. Cognitive constraints on compositional systems. *Contemporary Music Review*, 6(2):97–121, 1992.

[73] Vladimir Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, February 1966.

[74] Sol LeWitt. Paragraphs on conceptual art. *ArtForum*, 5(10):79–83, 1967.

[75] Sol LeWitt. Sentences on conceptual art. *Art-Language*, 1:11–13, 1969.

[76] Seth Lloyd. *Programming the Universe: A Quantum Computer Scientist Takes on the Cosmos*. Vintage Books, 2007.

[77] Abraham Moles. *Information theory and esthetic perception*. University of Illinois Press, 1966.

[78] Robert Morris. *Composition With Pitch-Classes: A Theory of Compositional Design*. Yale University Press, 1987.

[79] Taiga Nakamura and Victor Basili. Metrics of software architecture changes based on structural distance. In *Proceedings of the 11th IEEE International Software Metrics Symposium*, page 8, 2005.

[80] Mark Newman. The physics of networks. *Physics Today*, November 2008.

[81] Henri Poincaré. On the foundations of geometry. *Monist*, 9:1–43, 1898.

[82] Larry Polansky. Morphological metrics. *Journal of New Music Research*, 25:289–368, 1996.

[83] Larry Polansky, Alex Barnett, and Michael Winter. A few more words about James Tenney: Dissonant counterpoint and statistical feedback. preprint, 2009.

[84] Larry Polansky and Richard Bassein. Possible and impossible melody: Some formal aspects of contour. *Journal of Music theory*, 36(3):259–284, 1992.

[85] Tibor Radó. On non-computable functions. *The Bell System Technical Journal*, 41(3):877–884, May 1962.

[86] Jorma Rissanen. Generalized Kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, 20(3):198–203, 1976.

[87] Curtis Roads. *The Computer Music Tutorial*. MIT Press, Cambridge, Massachusetts, 1996.

[88] Curtis Roads. *Composing Electronic Music: A New Aesthetic*. Oxford University Press, forthcoming, 2010.

[89] Santonu Sarkar, Avinash Kak, and Girish Rama. Api-based and information-theoretic metrics for measuring the quality of software modularization. *Software Engineering, IEEE Transactions on*, 33(1):14–32, 2007.

[90] Santonu Sarkar, Avinash Kak, and Girish Rama. Metrics for measuring the quality of modularization of large-scale object-oriented software. *Software Engineering, IEEE Transactions on*, 34(5):700–720, 2008.

[91] Carla Savage. A survey of combinatorial gray codes. *SIAM Review*, 39(4):605–629, 1997.

[92] Claude Shannon. A mathematical theory of communication. *The Bell Systems Technical Journal*, 27:379–423, July 1948.

[93] Claude Shannon and Warren Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, September 1949.

[94] Mark So. text — composition scores and structure after 4'33. preprint, 2010.

[95] Ray Solomonoff. A formal theory of inductive inference, part i. *Information and Control*, 7(1):1–22, 1964.

[96] Brett Stevens, Mark Cooke, and Chris North. Beckett-Gray codes. preprint.

[97] Wilson Sutherland. *Introduction to Metric and Topological Spaces*. Oxford University Press, Oxford, England, 1975.

[98] James Tenney. John Cage and the Theory of Harmony. In Peter Garland, editor, *Soundings*, volume 13. Santa Fe: Soundings Press, 1983.

[99] James Tenney. *META + HODOS and META Meta + Hodos*. Frog Peak Music, Hanover, New Hampshire, 2nd edition, 1986.

[100] James Tenney. On 'crystal growth' in harmonic space. *Contemporary Music Review*, 27(1):47–56, 2008.

[101] Alan Turing. On computable numbers, with an application to the entscheidungsproblem. *Proc. London Math. Soc.*, 2(42):230–265, 1936.

[102] Duncan Watts and Steven Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, June 1998.

[103] Michael Winter. On James Tenney's *Arbor Vitae* for string quartet. *Contemporary Music Review*, 27(1):131–150, February 2008.

[104] Michael Winter and Azer Akhmedov. Generating chordal and timbral morphologies using Hamiltonian cycles. preprint, 2009.

[105] Stephen Wolfram. *A new kind of science*. Wolfram Media Inc., Champaign, Illinois, 2002.

[106] Iannis Xenakis. *Formalized Music: Thought and Mathematics in Composition*. Pendragon Press, Hillsdale, New York, 1971.

Appendix A

Scores

This appendix includes short descriptions and scores of the following pieces: *maximum change* (2007), *room and seams* (2008), *dissection and field* (2008), *small world* (2008), *towards completeness* (2008), *for Sol LeWitt* (2009), *for gregory chaitin* (2009), *field and perfect circuit* (2009), *recitation, code, and (perhaps) round* (2009), *Approximating Omega* (2010) and *pedal, triangle machine, and (perhaps) coda* (2010). Their inclusion intends to provide concrete examples of how music can inform research and vice versa.

Note that several of these works integrate ideas expressed in *Structural Metrics* on several hierarchical levels. For example, many of the works actually define subsets of possibilities (as discussed in the preface) of a meta-idea first defined in a piece called *sound.sound* (2007). In this piece, the title is the score itself. In the pieces that subset *sound.sound*, the realization of the mathematical objects that are the '.' in *sound.sound* are singular occurrences that are ephemeral with respect to the entirety of the piece (as opposed to a 'grander scheme' in which all things are ephemeral). This idea of ephemera

prompted the more general discussion of ephemera provided in the conclusion. But what makes something ephemeral, especially in a temporal soundscape such is the case with music, is that its existence be brief and fleeting. In most of these pieces, the duration of the work is undetermined. The duration of the realization of the '.' is most often expressed as a proportion of the length of the entire piece. This is possible by the structural definition of scalable relations also discussed in the conclusion.

On the other hand, works such as *maximum change* and *small world* consist exclusively of one idea that permeates the entire piece: the realization of a particular mathematical object.

A.1 *maximum change*

maximum change for four percussion instruments is the first piece in the appendix which engendered ideas for *Structural Metrics*. The piece enumerates all timbral possibilities of a static chord with four pitches. Each instruments can play up to all four pitches at once. The local morphological constraint is that from chord to chord, each pitch is sounded by a different instrument (hence the title *maximum change*). The global morphological constraint is that each chord sounds only once.

The morphology can be mathematically defined and abstracted as a Hamiltonian path on a graph denoted as $T_{n,k,l}$ where l, k, n are natural numbers satisfying the condition $l \leq k$. The set of vertices of $T_{n,k,l}$ consists all k-tuples $(x_1, x_2, \ldots, x_k)$ where $x_i \in \{1, 2, \ldots, n\}$. Two such k-tuples $\alpha = (x_1, x_2, \ldots, x_k)$ and $\beta = (y_1, y_2, \ldots, y_k)$ are connected by an edge if and only if $x_i = y_i$ for exactly l values of $i \in \{1, 2, \ldots, k\}$.

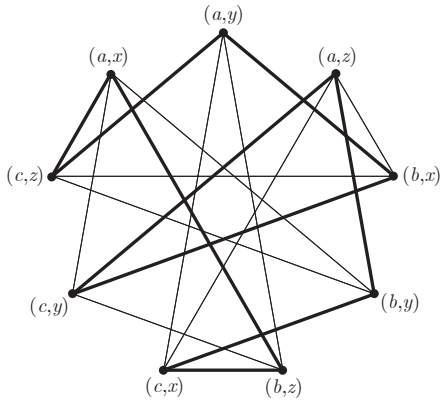


Figure A.1: Graph of $T_{3,2,0}$ with a Hamiltonian cycle

In the case of *maximum change*: $n = 4$ is the number of symbols or timbres; $k = 4$ is the tuple or chord size, and $l = 0$ is the number of positions in the tuple that have the same value from chord to chord (or tuple to tuple or vertex to vertex). For example, a vertex represented by $\{1, 2, 3, 4\}$ denotes that C, D, E, F-sharp may be performed by crotales (1), glockenspiel (2), chimes (3), and piano (4), respectively. Since the local morphological constraint is that each pitch be performed by a different instrument from chord to chord, then the vertex representing $\{1, 2, 3, 4\}$ is connected to $\{2, 1, 1, 1\}, \{2, 3, 2, 2\}$, etc. Since the graph is Hamiltonian, a non-repeating, exhaustive enumeration satisfying the local morphological requirements is possible. In [104], it is shown that all graphs $T_{n,k,l}$ where $n \geq 3$ and $k \leq l + 1$ are Hamiltonian.

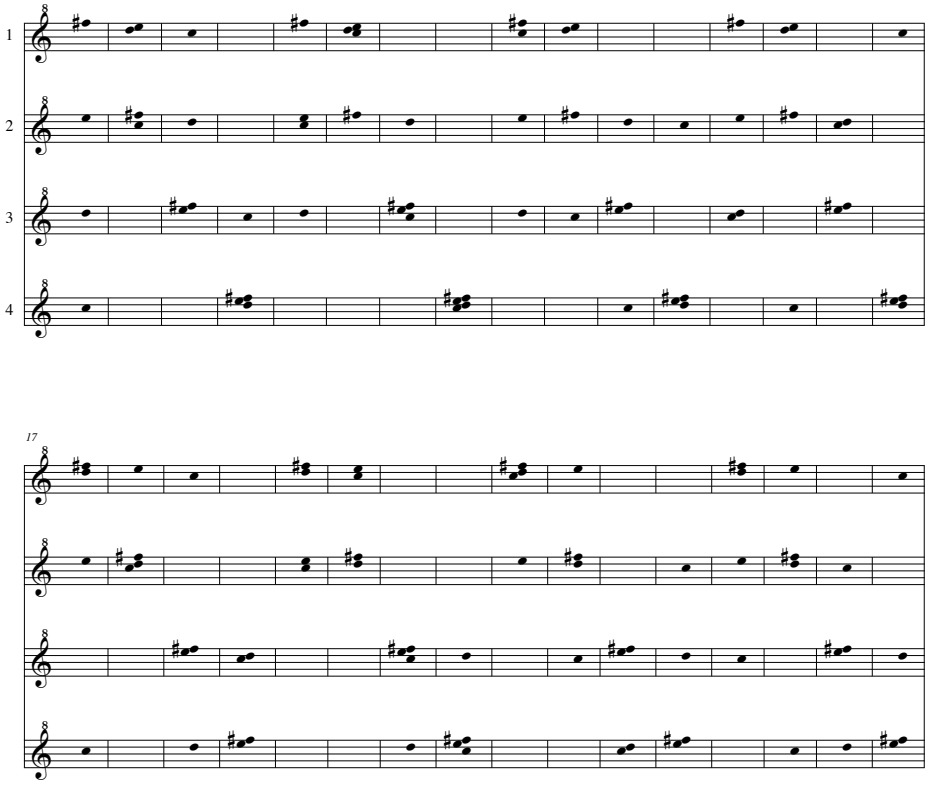
maximum change
for azer akhmedov

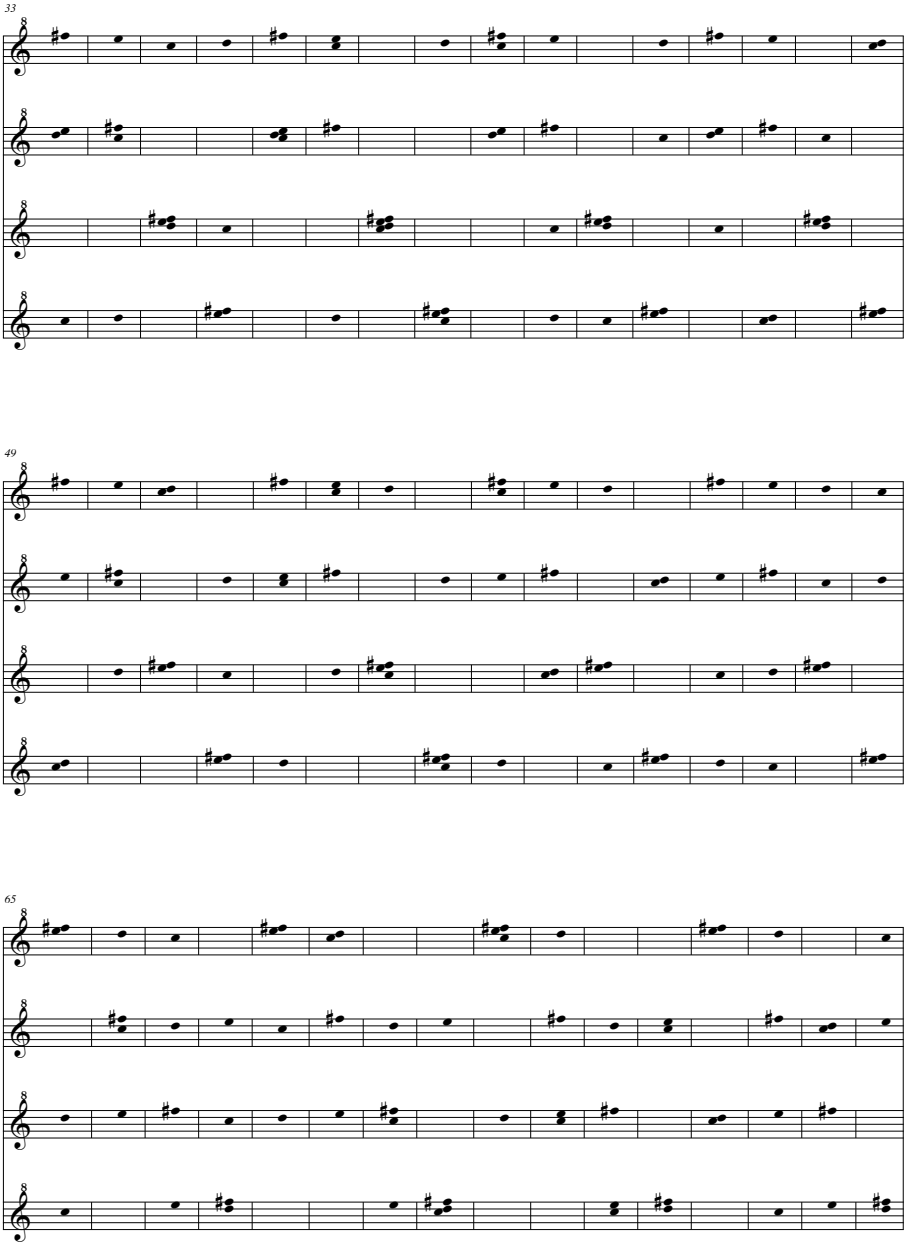
dedicated to the people of the world against whom a war on civil liberties, democracy, prosperity, and sovereignty is currently waged by a small, elitist ruling class intent on global power. may we overcome such tyranny and live in peace.

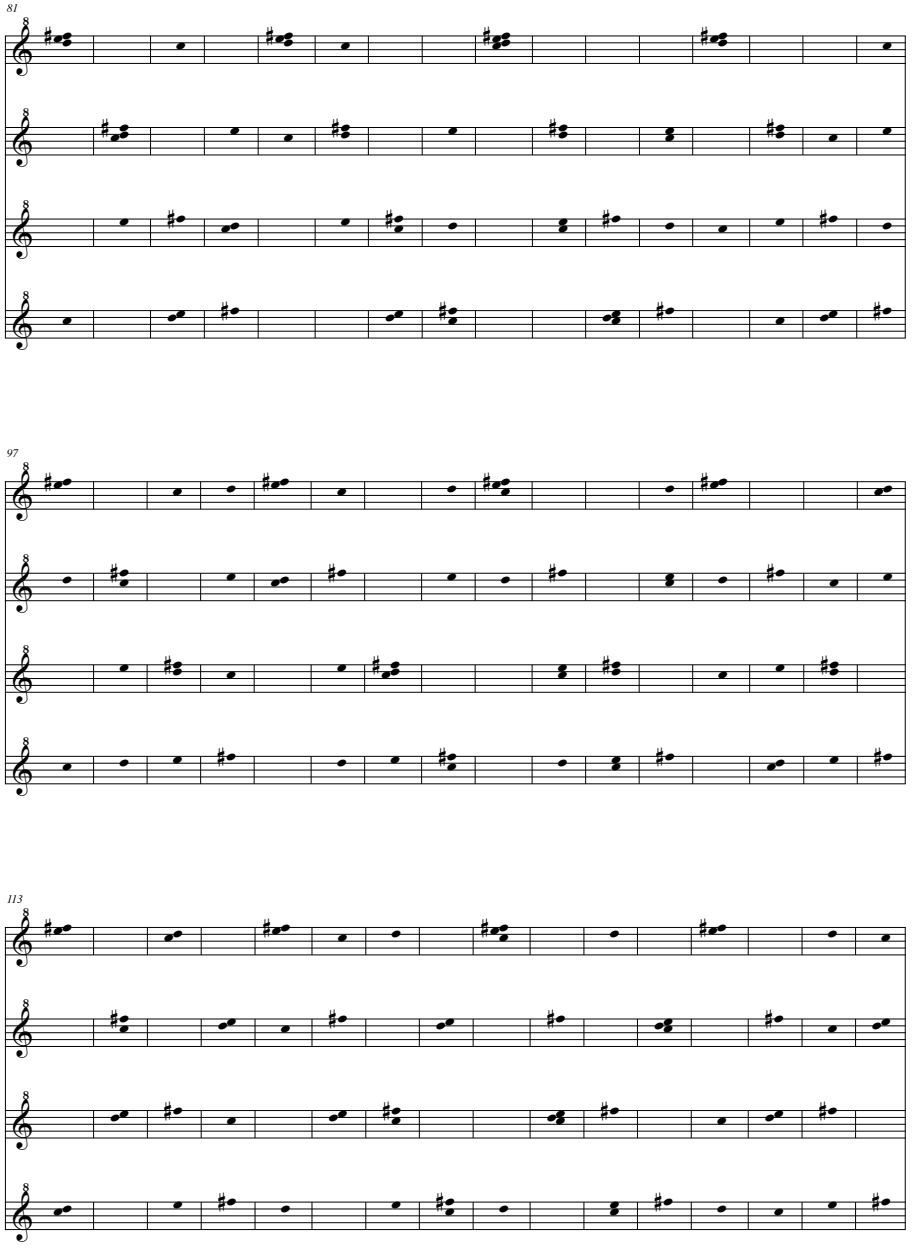
-mike winter (2007)

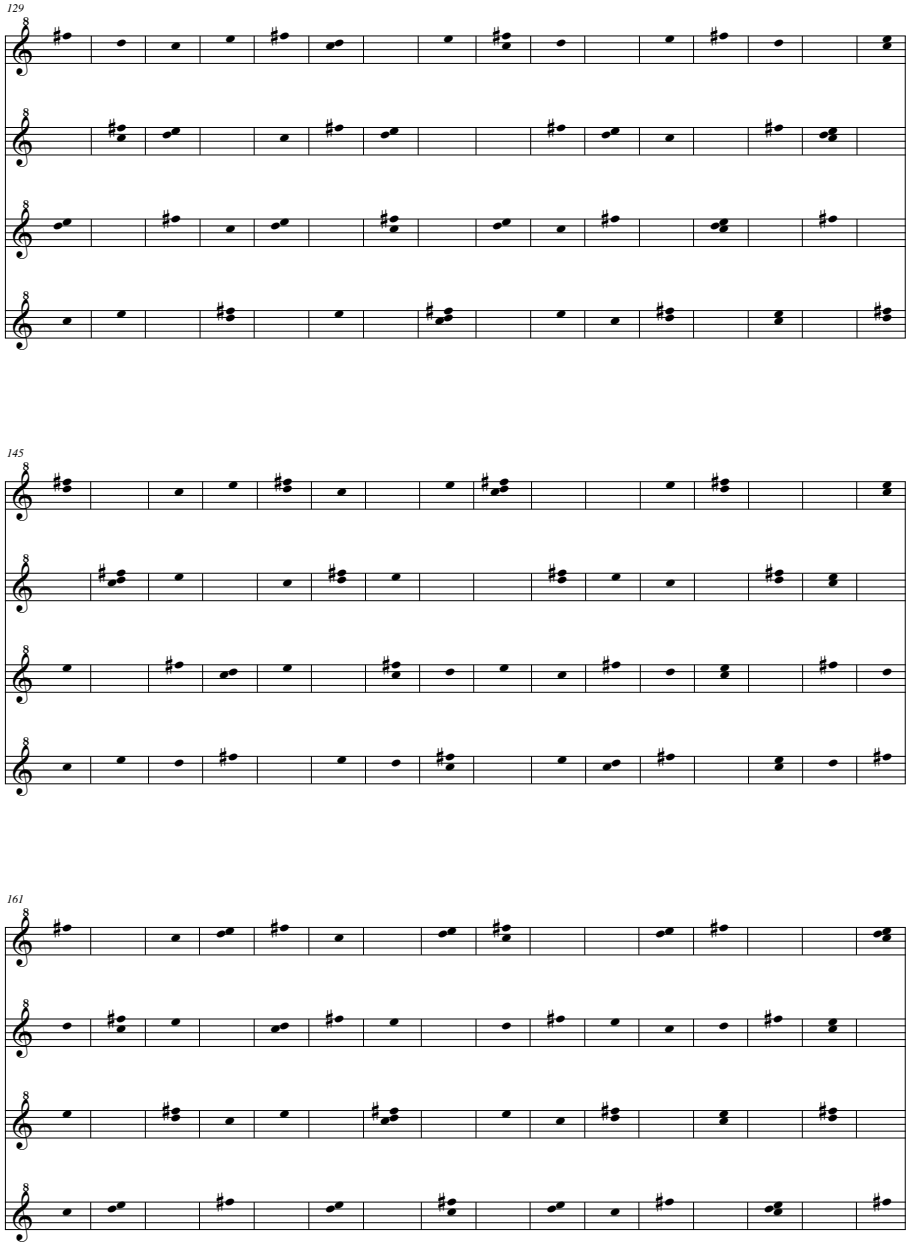
- to be performed with four percussion instruments of long decay with clearly distinct timbres. for example, circular plates, rectangular bars, hollow tubes, and struck strings such as crotales, glockenspiel, chimes (with c_6, d_6, e_6, and $f^{\sharp}_6$ removed from a set such that they can be struck individually and together), and piano (struck with mallets inside the piano with the sostenuto pedal depressed throughout keeping the dampers off c_6, d_6, e_6, and $f^{\sharp}_6$), respectively. each instrument placed reasonably far apart from the others.
- a pitch in the given score can be transposed such that every occurrence of that pitch is also transposed the same amount and such that the conglomerate chord always consists of four different pitches. for example, all c_6 can be changed to b_5.
- each tone in a given measure struck simultaneously across all parts and allowed to ring freely.
- eight to twelve seconds between each attack.
- a constant strike velocity throughout such that all tones sound clearly and equally present but not loud.
- score is written as sounds.

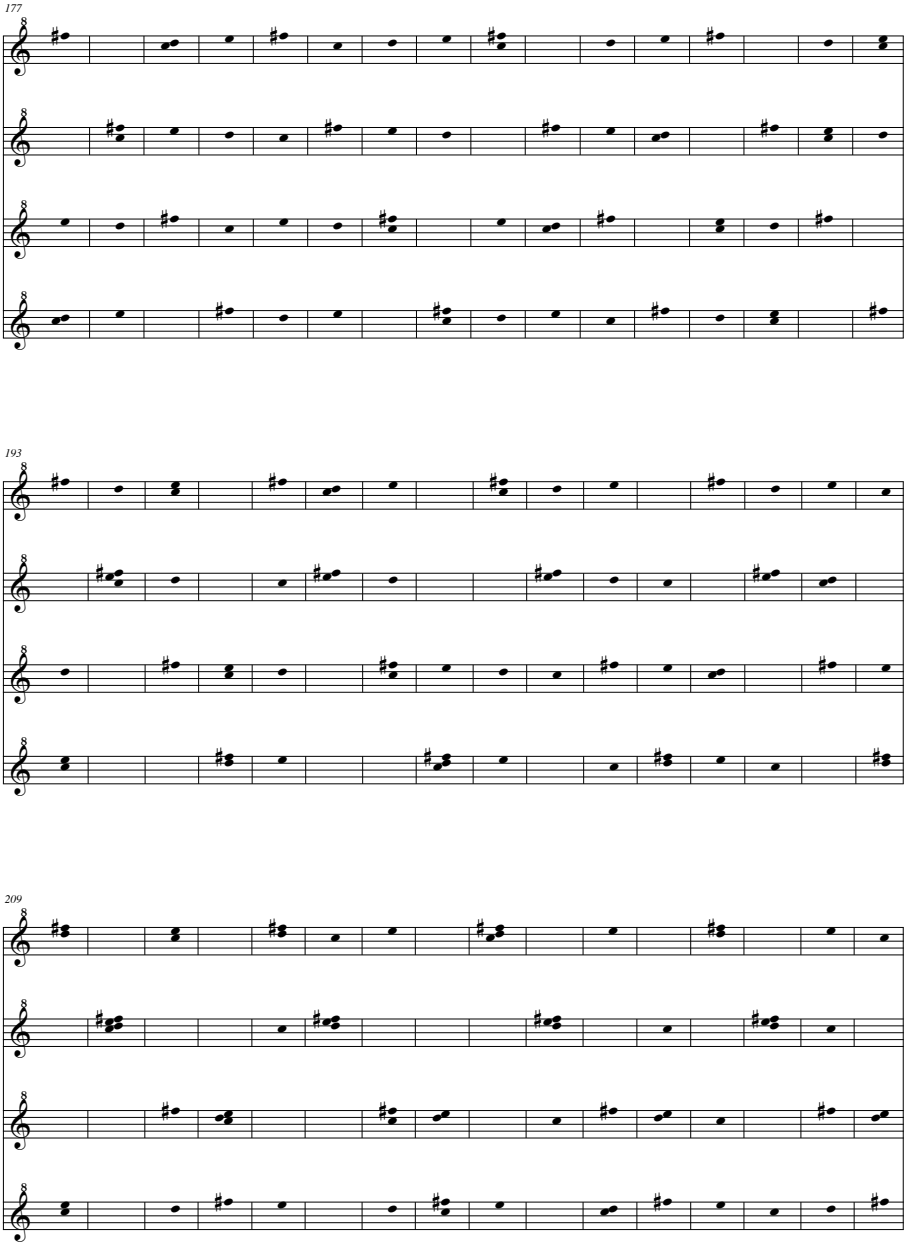
a special thanks to azer akhmedov whose help made this work possible.

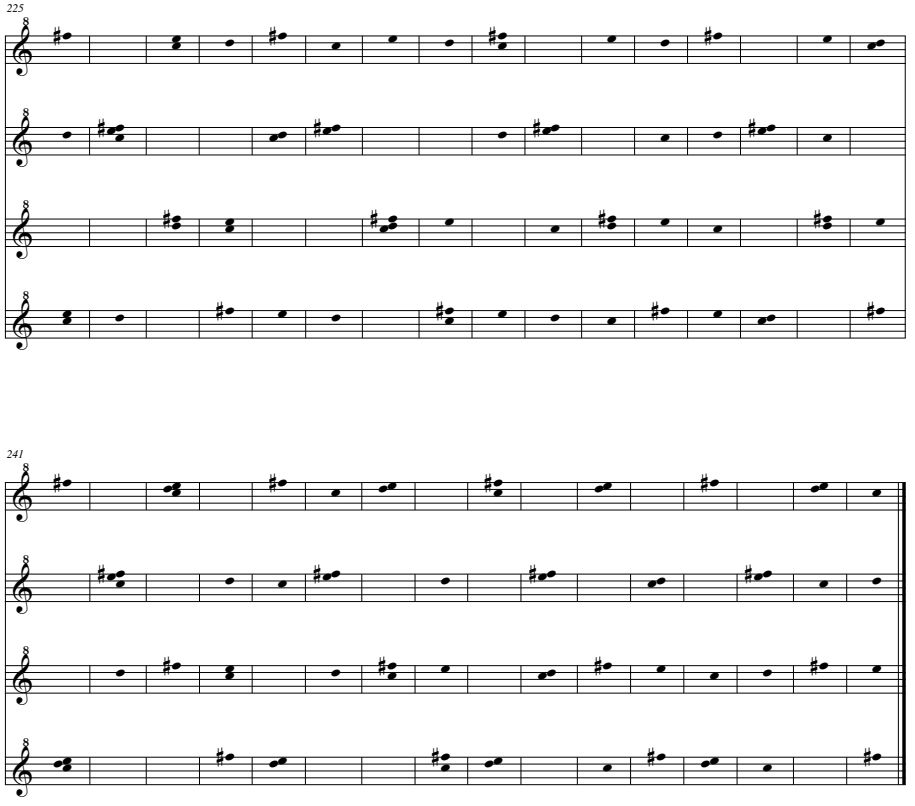



33
49
65


81
97
113


129
145
161


177
193
209


225
241

A.2 *room and seams*

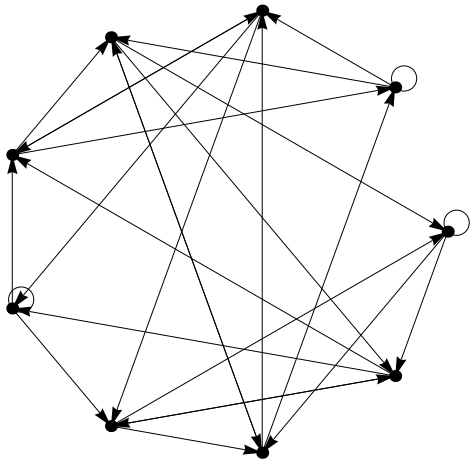


Figure A.2: de Bruijn graph of $B(3, 2)$

room and seams is constructed from a mathematical object called a de Bruijn sequence (after Nicolaas Govert de Bruijn [43]). A de Bruijn sequence, $B(n, l)$, contains all words of length l from an alphabet of size n such that every word appears only once because it overlaps with the previous and next words by $l - 1$ characters. Essentially, it is the fastest way to brute force a combination lock. The sequence can be represented by a Hamiltonian path on a directed graph (often referred to as a de Bruijn graph) where the vertices are all words of length l from a given alphabet of size n and two vertices are connected by a directed edge if the in-vertex overlaps by $l - 1$ characters with the out-vertex (see Fig. A.2).

In *room and seams*, the alphabet size and word length equal 4. Each character in the alphabet represents a group of performers located in a room as far as possible from all other groups. When a group's character occurs in the sequence, they sound a tone.

Thus, the tone, which is of the same pitch for all performers, is passed around the room such that every spatial sequence of length 4 occurs in the shortest total morphology possible.

room and seams
for l. taylor and n.g. de bruijn

each staff is read by a different group of people. each group situated in a room as far from the others as possible.

a group consists of at least one person playing a pitched percussion instrument that when struck and allowed to ring has a long decay time or can also be struck to make a short punctuated sound. the group may also include instruments that can sustain a tone continuously. the sustained tone instruments need not play every measure.

each measure indicates a span of time that is variable from measure to measure. changes in time per measure should be statistically uniform in general though they may still vary a great deal.

all performers play the same pitch throughout, which is also the same for all groups. notes in different staves within the same measure should be interpreted with simultaneous attacks.

for the percussion instruments, unfilled notes represent tones that should be allowed to ring until the next tone is initiated or until the natural decay of the instrument, whichever comes first. for sustained tone instruments, each tone should be held for at least the duration of the percussion tone's decay and at most the duration till the next initiated tone. if there is more than one sustained tone instrument in a group, exits may be staggered. any duration of silence may occur before a new tone is initiated. the performers are encouraged to explore no silence or silence equally.

filled notes with a dot above represented a very short punctuated tone performed only by the percussionist.

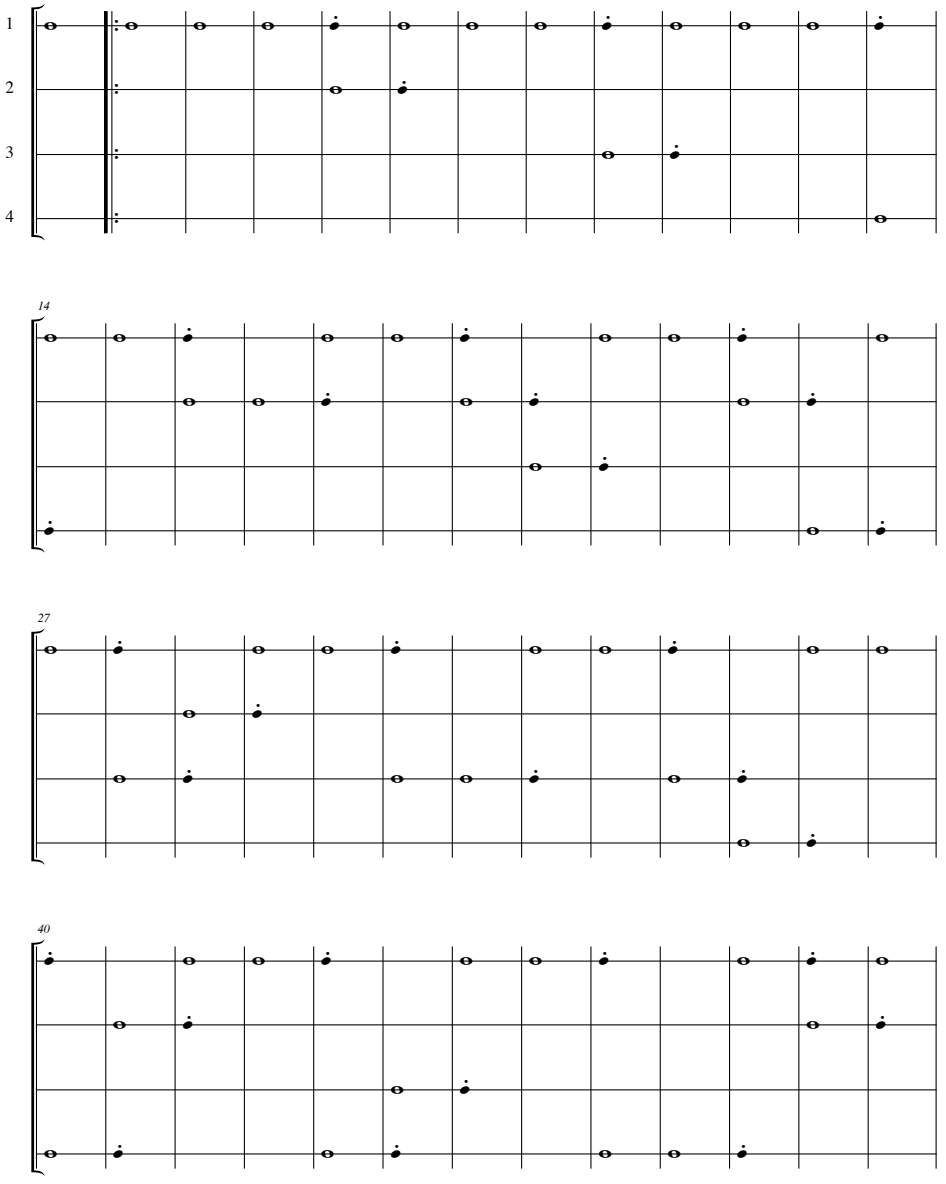
repeats optional. any number of times.

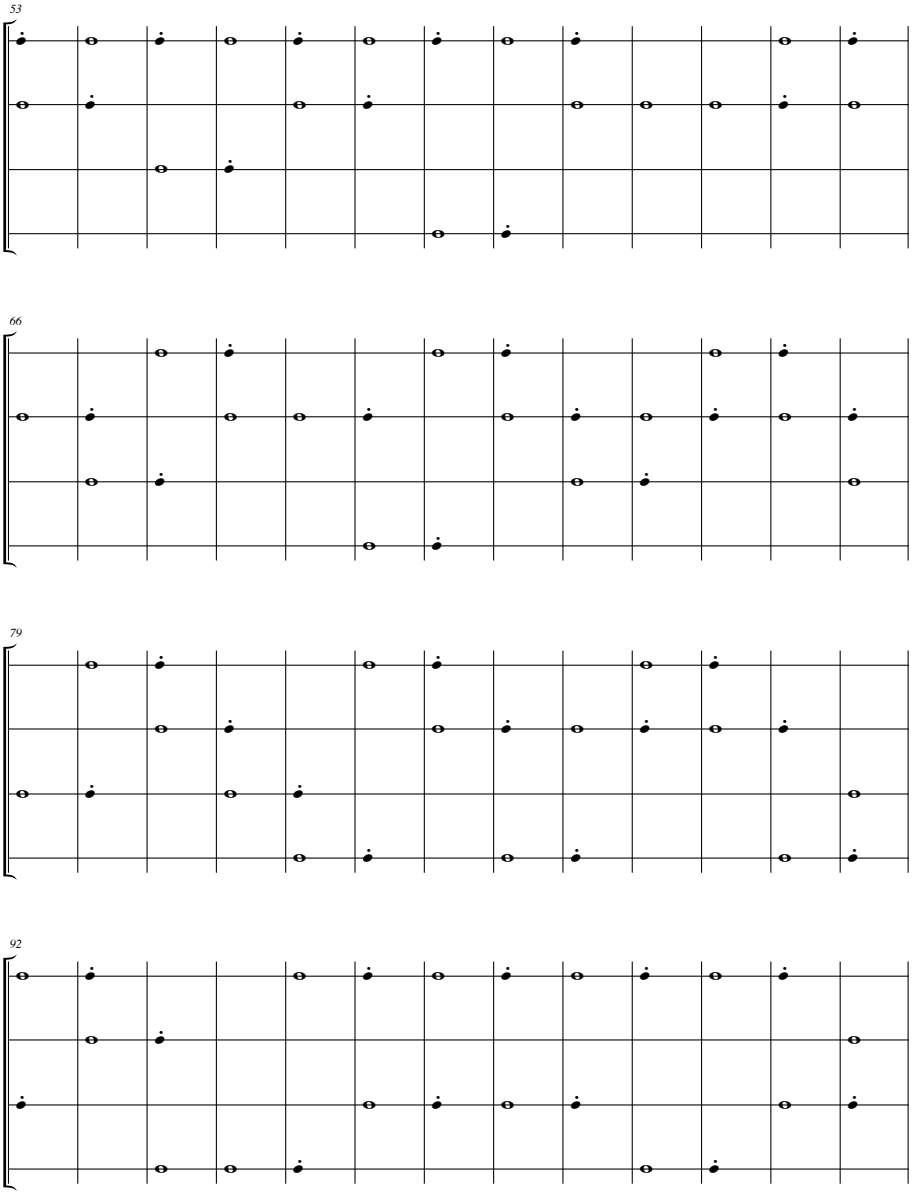
dynamic generally constant.

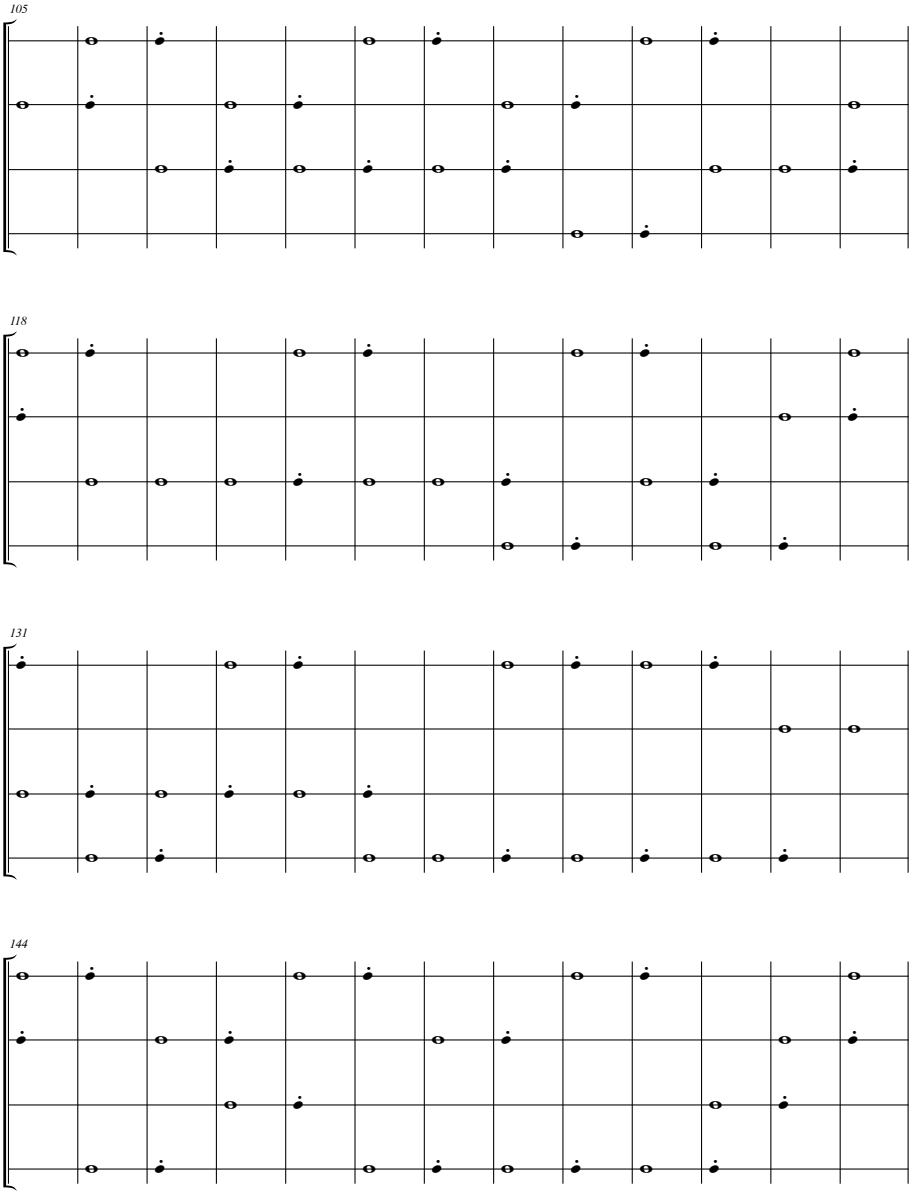
clear but not loud.

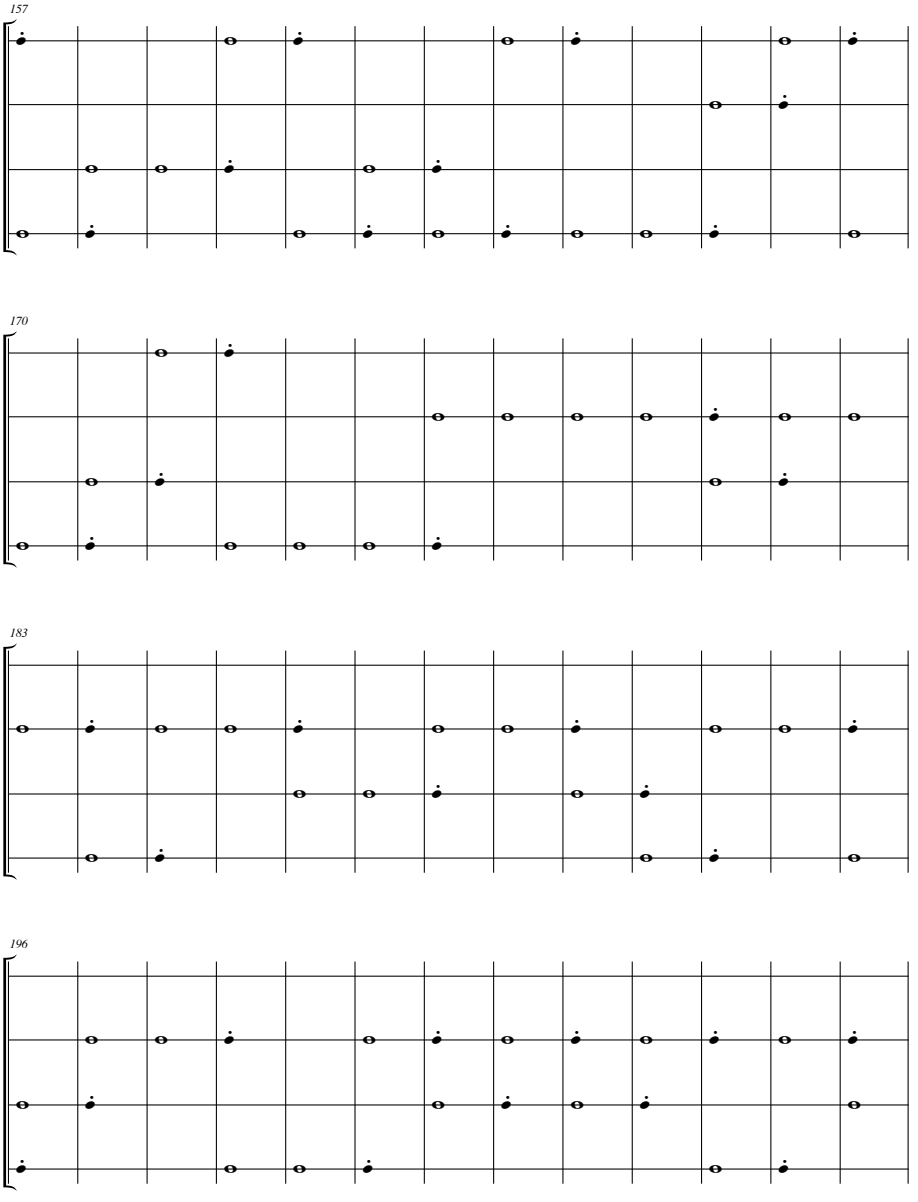
michael winter (june 2008)

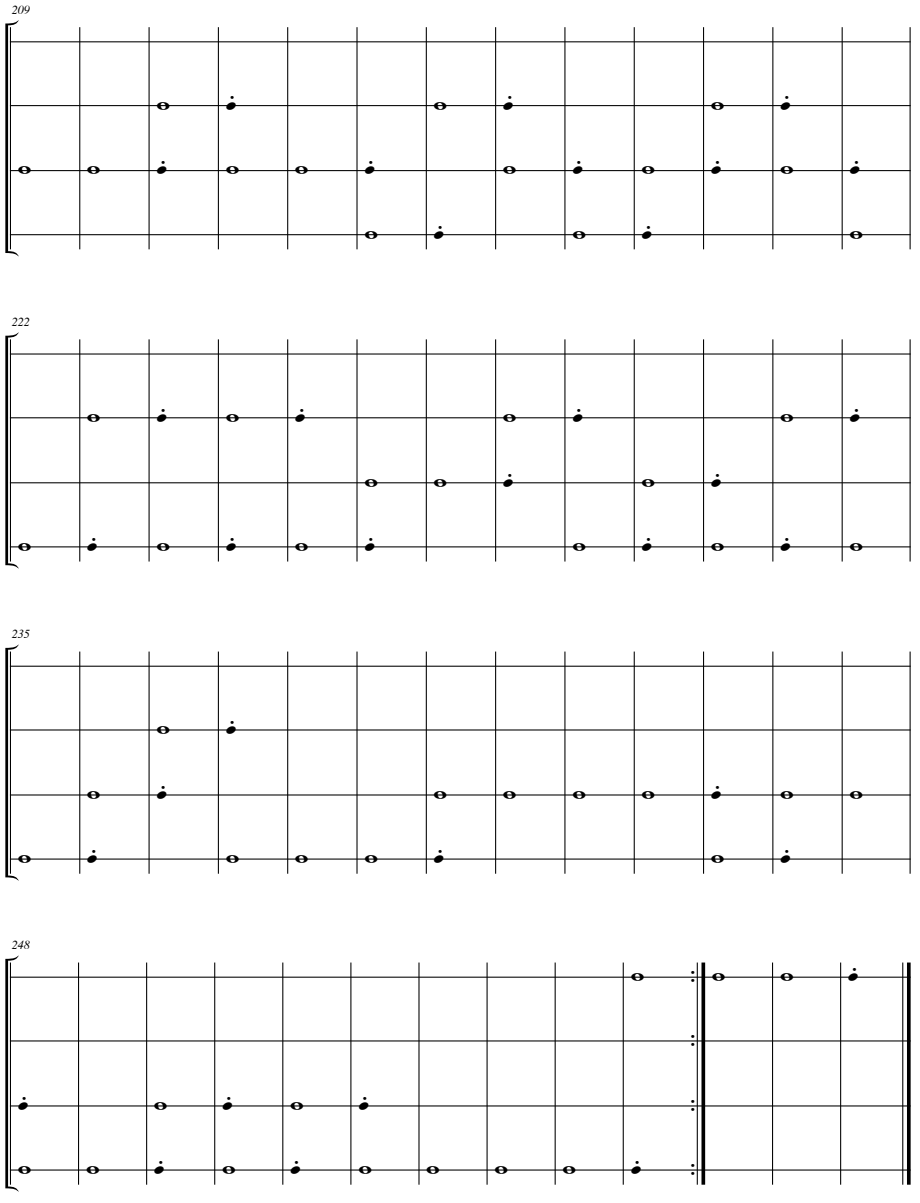
room and seams




53
66
79
92


105
118
131
144


157
170
183
196



michael benjamin winter (2008)

5

A.3 *dissection and field*

As with *room and seams*, *dissection and field* is also constructed from a de Bruijn sequence. The alphabet in this case consists of three numbers $\{-1, 0, 1\}$ that represent direction in a pitch morphology; down, same, up, respectively. The word length is 6. The piece goes through all pitch contours of size 6 in as few tones as possible. There is an added global morphological constraint that the running sum of the sequence stay within a reasonably small range so that pitches do not get extremely high or low. A computer program in a constraint programming language generated such a sequence. The final pitch morphology was pieced together from several musical fragments such that it conformed to the contour sequence.

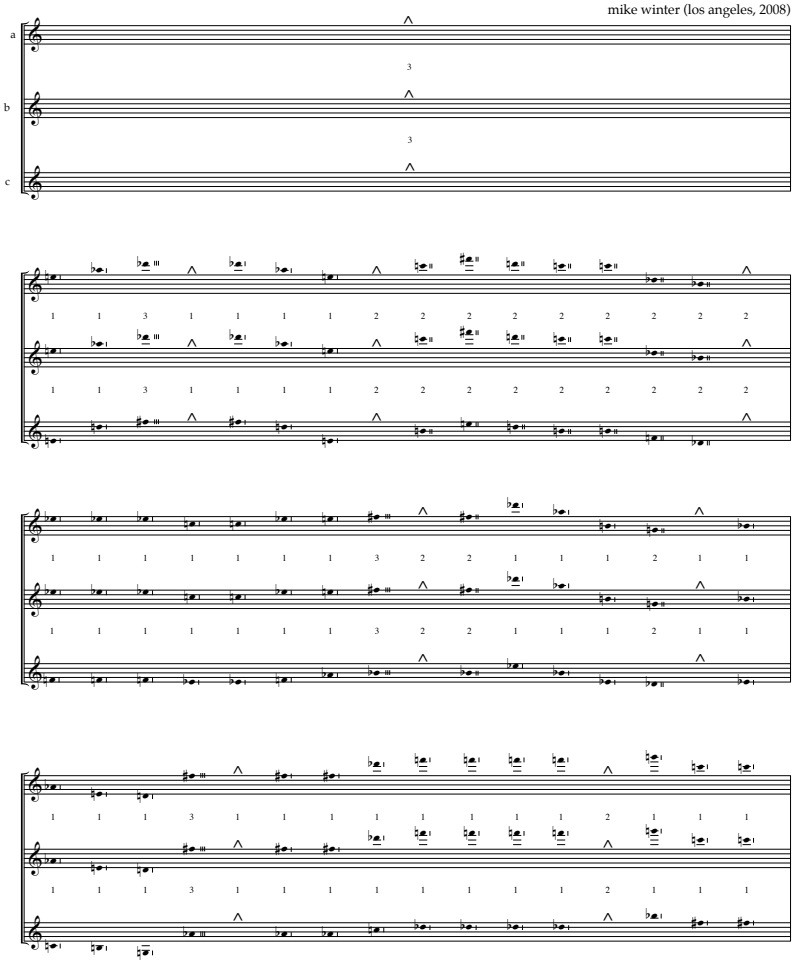
The piece integrates the above idea with larger formal concerns. Two groups run in parallel following the contour sequence. In the score, the notes and rests (the latter of which were arbitrarily inserted) have associated markings that indicate general durations. Each performer plays independently of the others, which blurs the sequence to some extent. Also, one of the performers departs from the sequence for a significant portion of the piece and sustains a high-pitched tone.

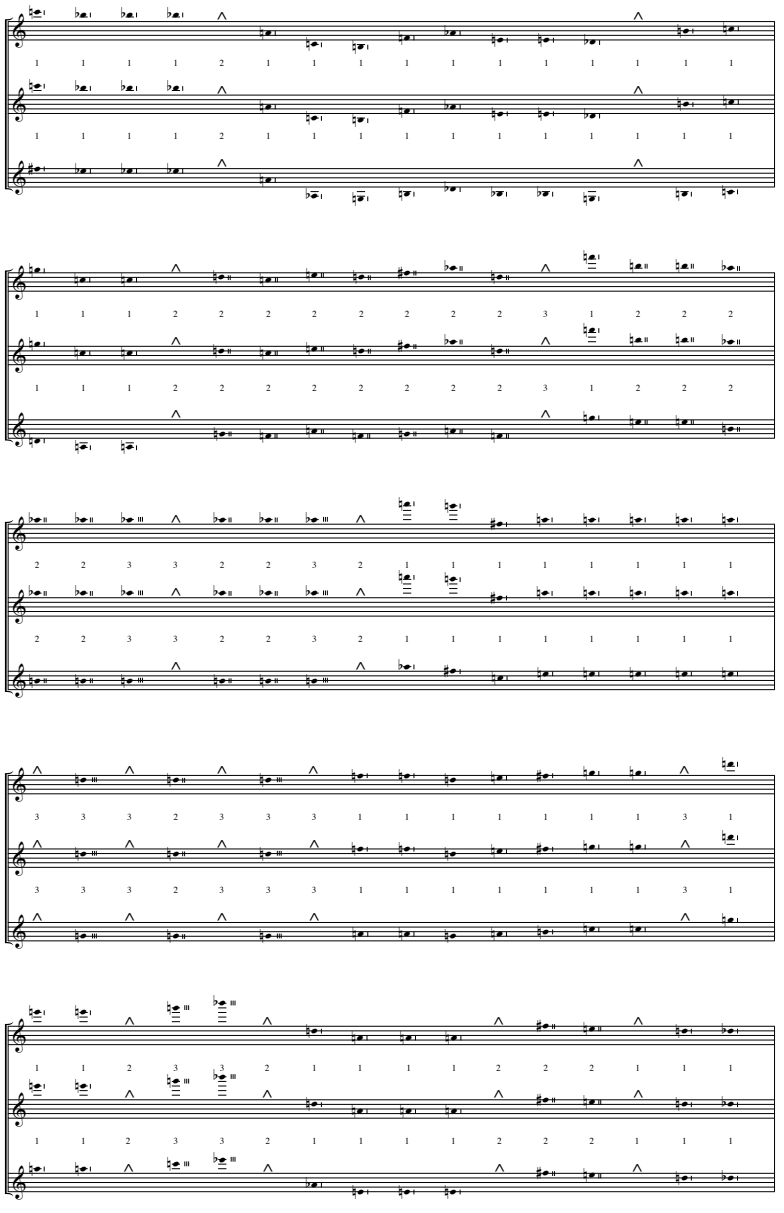
dissection and field

- for miwako abe
- dedicated to hakan kjellerstrand and christian wolff
- in memoriam harris wulfson

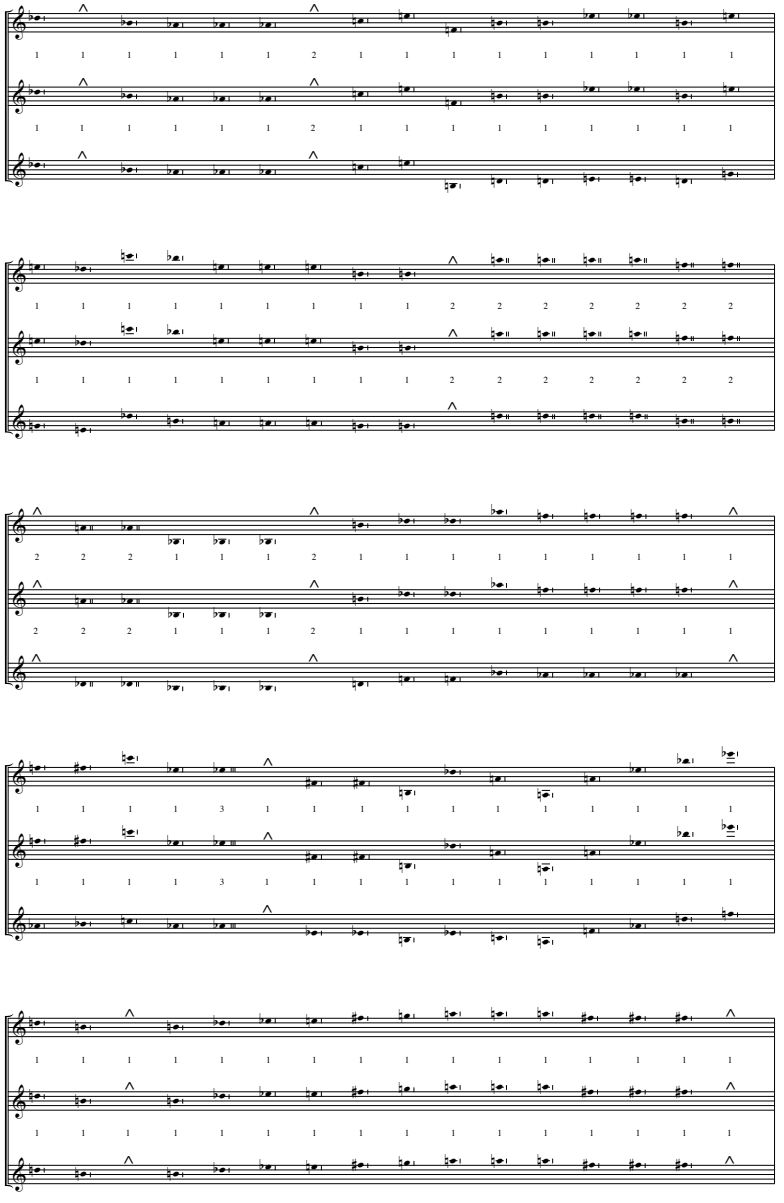
sometimes together, sometimes apart. one or more performers per part/staff such that all notes are realized, but not necessarily by all performers. numbers above or below each staff indicate general duration for each sound or silence. all 2s indicate longer durations than all 1s and all 3s longer than all 2s. some 3s should be realized as very long. the number of tick marks appended to each notehead corresponds to the number indicating general duration. the caret symbol indicates a rest (see christian wolff's exercises). if the ensemble is mixed with instruments that cannot sustain the same length (such as winds and/or percussion with strings), long tones may be held (much) longer by the performers that can sustain longer. in this case, the performers that cannot sustain as long wait in silence accordingly after finishing the tone. between rests, repeated numbers are realized as the same duration. successive notes uninterrupted by a rest may be slurred. performers should explore pitch deviations between 50 cents above and below the written pitch. a high triangle notehead without ledger lines indicates a high tone with a pitch that is the same between the top two parts (though not necessarily the highest pitch possible or the highest pitch in the piece). this tone sustains continuously in the topmost part for a long time. the entrance and exit of this tone may be accented by a pitched percussion instrument with a decay of half a second or more. any note in a part may be transposed such that—except for the sustained high tone explained above—the contour of successive notes remains the same (up, down, sameness), vertically aligned notes in the top two parts remain in unison, and notes in the bottom part remain lower than or equal to vertically aligned notes in the top two parts. throughout; soft, yet clear.

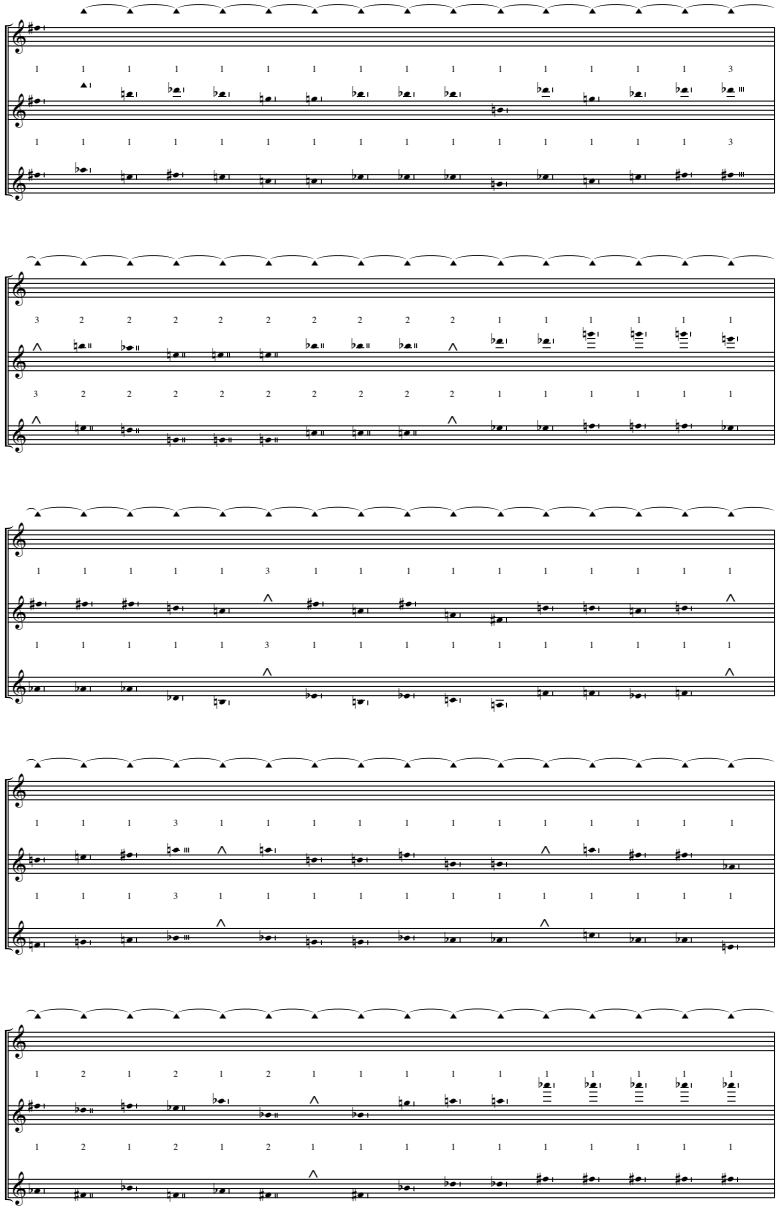
mike winter (los angeles, 2008)



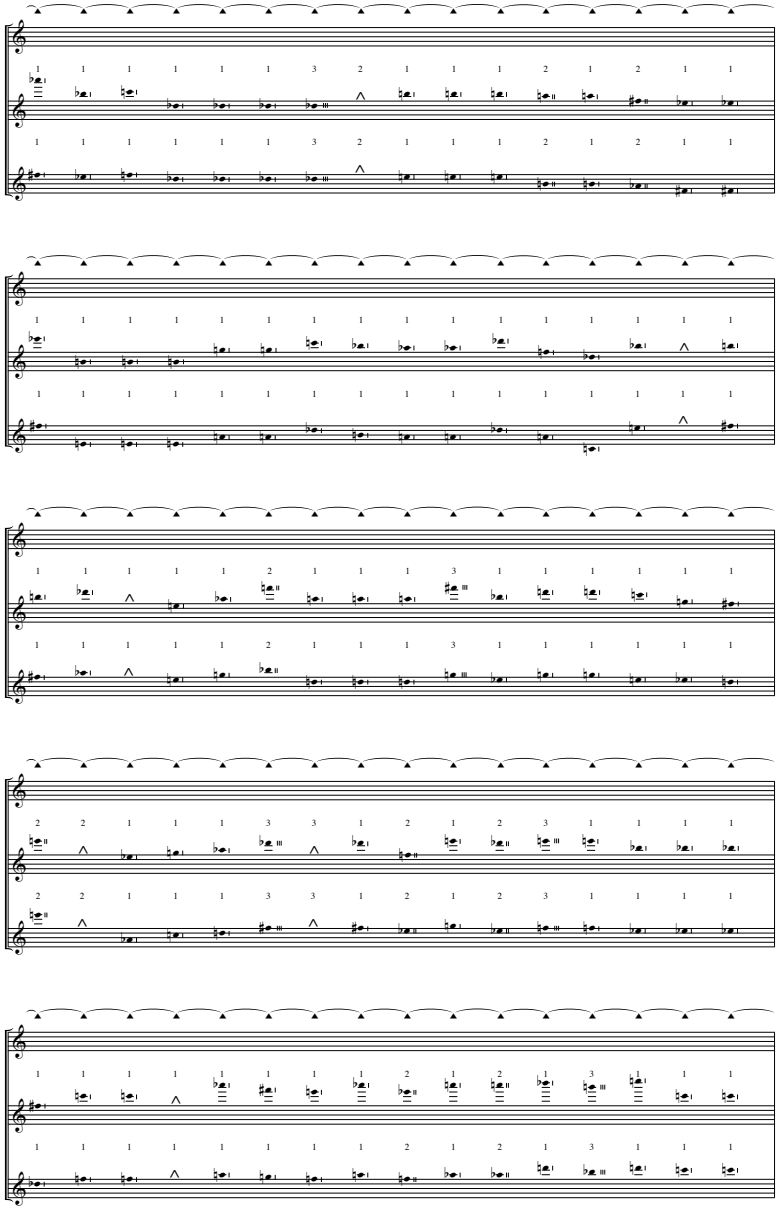


2

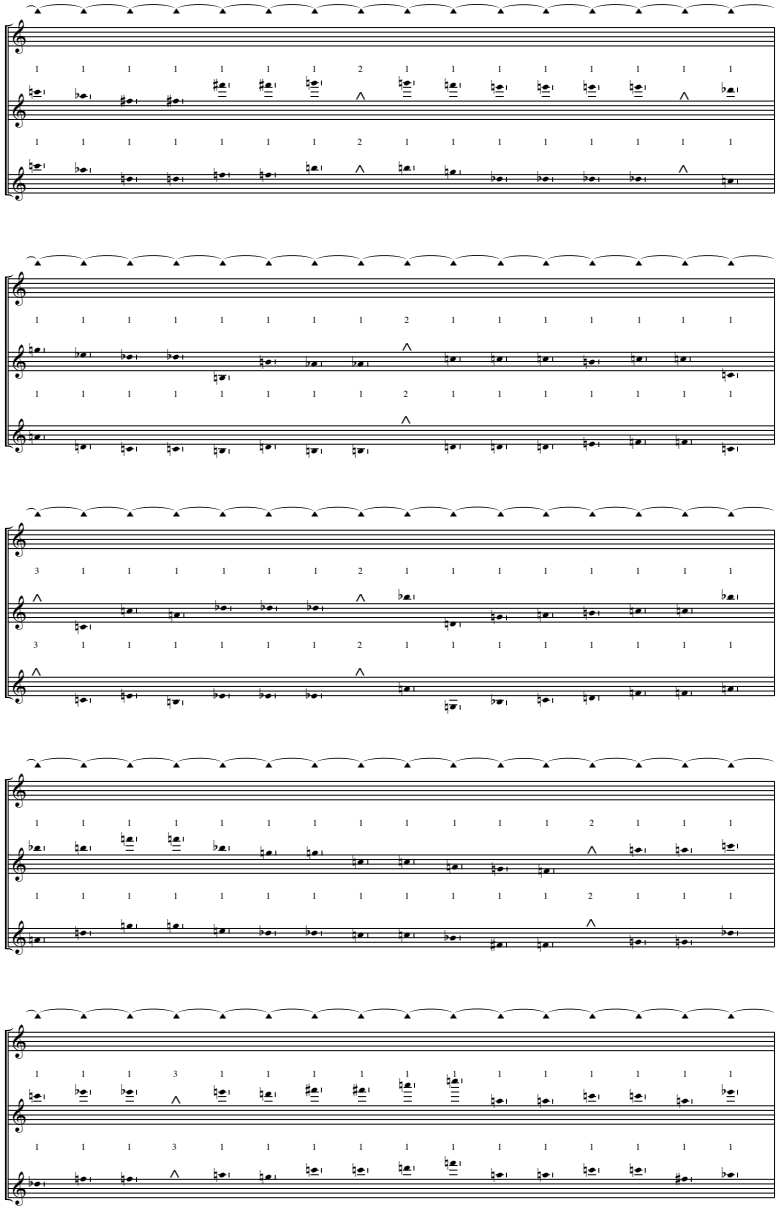




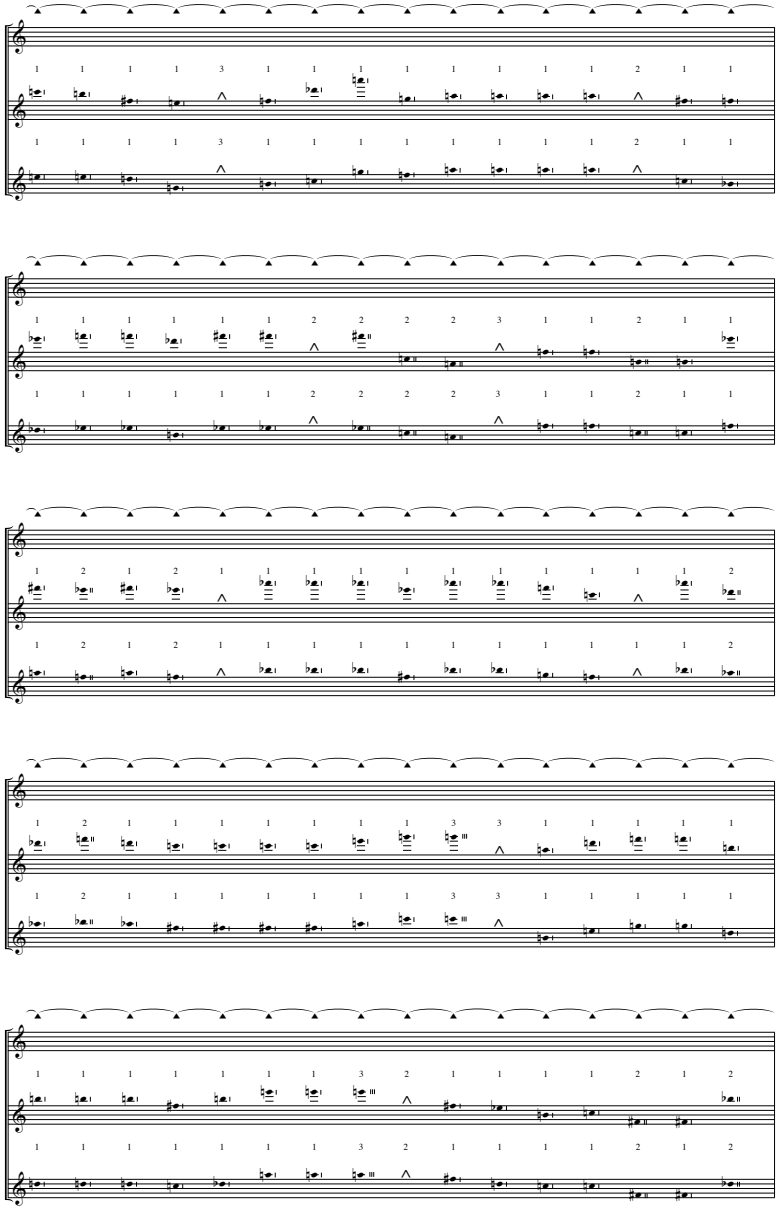
4

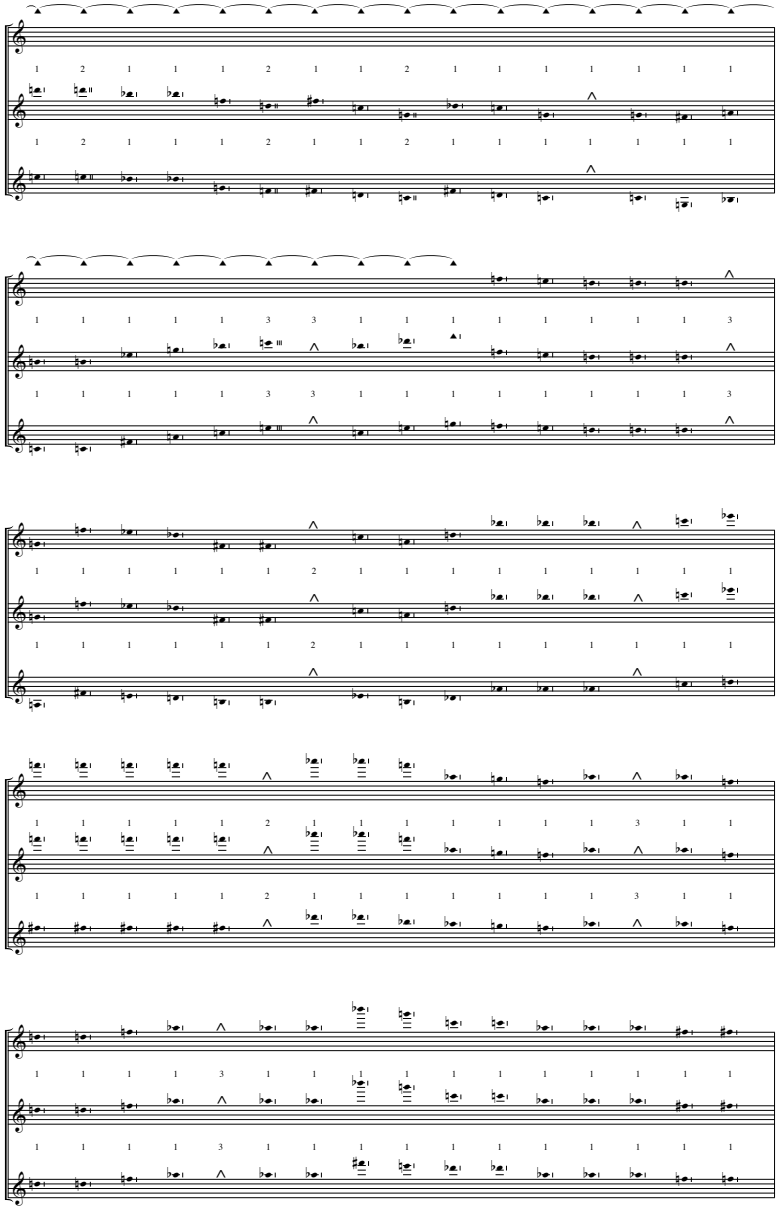


5

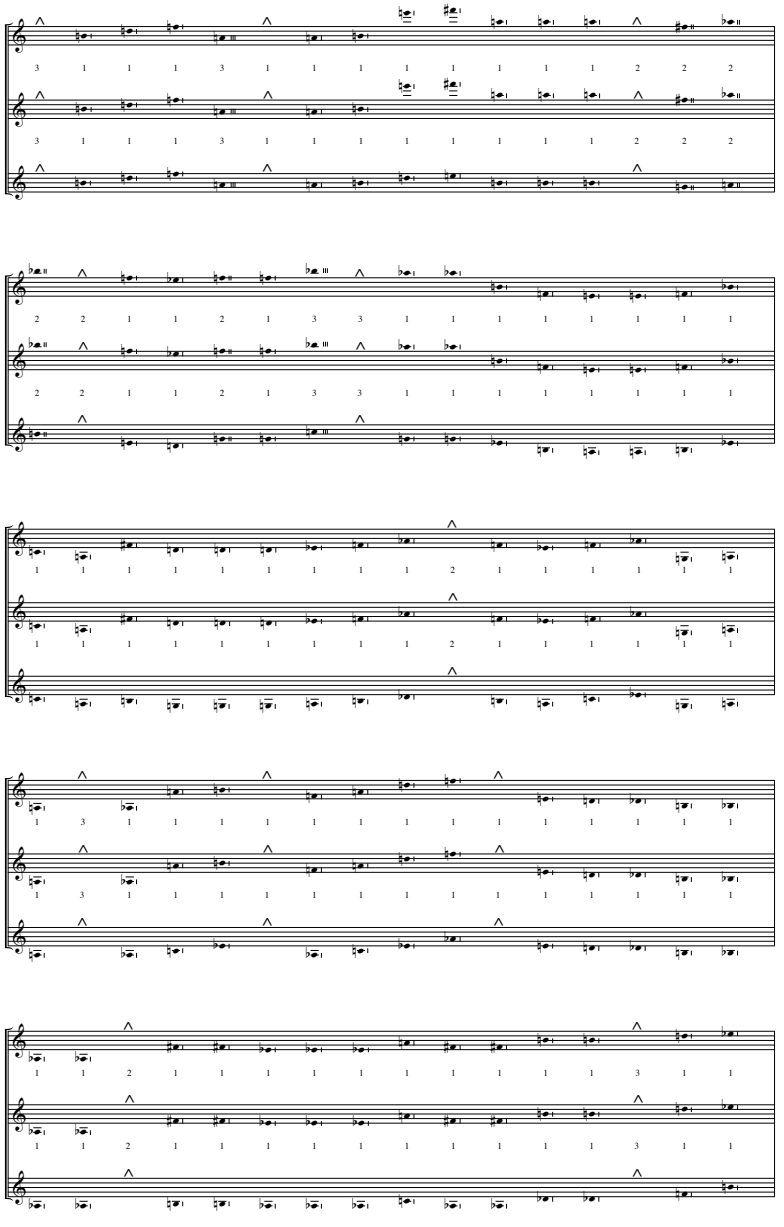


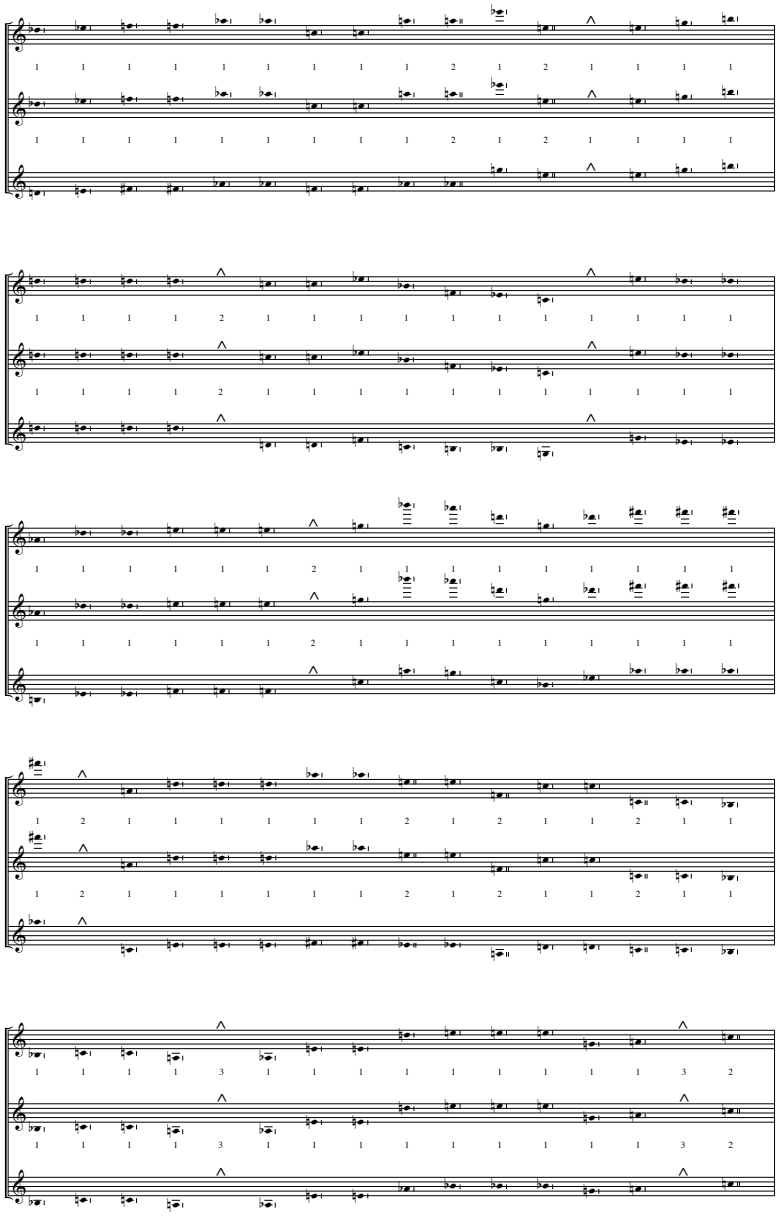
6



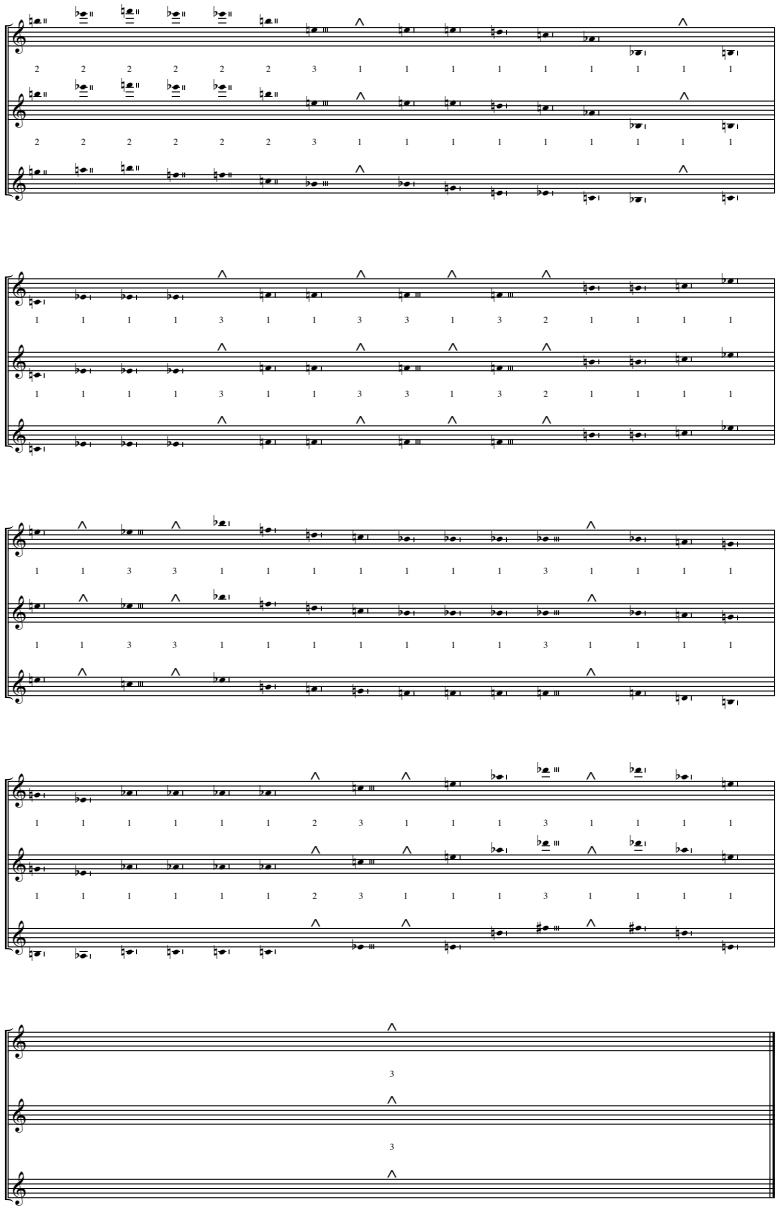


8





10



11

A.4 *small world*

small world is an exploration in structural isomorphism. Players independently interpret a randomly generated, connected graph where vertices represent events and edges determine whether or not an event can succeed another on an arbitrary walk on the graph. The events, which are variable, are predefined by the ensemble. The piece is therefore predicated only on structure and not materiality. The performers create linear morphologies with temporal dependencies by exploring paths in that structure. In this sense, *small world* is also an homage to Christian Wolff's *For One, Two or Three People*. For different performances that realize the same graph, the structure remains the same regardless of the events defined by the ensemble. Note that the implemented graphs are not necessarily "small-world networks" as outlined in [102], rather the title is simply a play on the notion of connectivity.

small world

a random, connected graph with v vertices and e edges such that $5 \leq v \leq 79$ and $v - 1 \leq e \leq 2v$. each vertex represents a pre-determined, distinct event. for example, an event might be a fragment or event of another piece by other composers including, but not at all limited to, joseph kudirka, johannes ockeghem, james orsher, michael pisaro, mark so, tashi wada, christian wolff, etc. or the events may be differentiated by timbre, pitch, dynamic, duration, location, type of action or sound object, etc. perhaps explore events that may create an impasse. for example, assigning a vertex an event that must interact with an already occurring event or that can only occur after the occurrence of an event or succession of events within or without the set of events assigned to the vertices of the graph (see christian wolff's *for one, two, or three people*).

a starting vertex (or vertices for several voices; a voice being an individual path through the graph) is arbitrarily determined for the first occurring event of each respective voice (though each voice need not enter simultaneously). each successive event within a voice must be an event assigned to a neighboring vertex of the previous vertex/event. the exploration of voice following in moderation is encouraged

the piece ends any time after each voice has played all the events or when an impasse has been reached.

---

the following pages contain an example set of 20 events. also, a series of graphs (randomly generated by the first example algorithm provided below) all with 20 vertices but each with a different number of edges are provided. the graphs have been formatted for readability. distances between vertices do not indicate anything such as duration. all that matters is whether two vertices are connected implying a temporal succession. these examples are freely usable and changeable. it is also encouraged to create different graphs and event sets and to realize more than one graph simultaneously

---

a random, connected graph may be generating using the following algorithms.

example 1:

1) choose a number of vertices, v, and a number of edges, e, such that $5 \leq v \leq 79$ and $v - 1 \leq e \leq 2v$;
2) initialize the graph by adding two vertices and connecting them;
3) add a vertex and connect it to a vertex already present in the graph;
4) repeat step 3 until v vertices are present in the graph.
5) choose 2 random vertices to connect such that the edge does not create a loop or a parallel edge;
6) repeat step 5 until the e edges are present in the graph.

    • note that steps 5 and 6 are not necessary if $e = v - 1$.

example 2:

1) choose a number of vertices, v, and a number of edges, e, such that $5 \leq v \leq 79$ and $v - 1 \leq e \leq 2v$;
2) generate all connected graphs with v vertices and e edges;
3) randomly select one of the graphs.

-mike winter (los angeles; september 2008)

1) successive subsets of a harmonic series; sounded melodically descending or ascending

2) stones; soft

3) a relatively loud burst of short percussive sounds followed by an indefinite duration of silence

4) many, many very small falling objects

5) a long continuous tone after at least 10 seconds of silence

6) the melody of a folk song with several notes replaced by silence or transposed by a small amount

7) several sine tones of random pitch with staggered, imperceptible entrances and exits

8) a very slow glissando

9) 10 to 20 seconds of a sound similar to white noise

10) a tangible gift; made during the piece; presented or displayed

11) a very slow glissando that intersects with a glissando already sounding

12) a resonator imposed over and activated by an already sounding event

13) a series of words generated from a markov chain of some high order

14) waiting; waitfulness

15) a distant sound

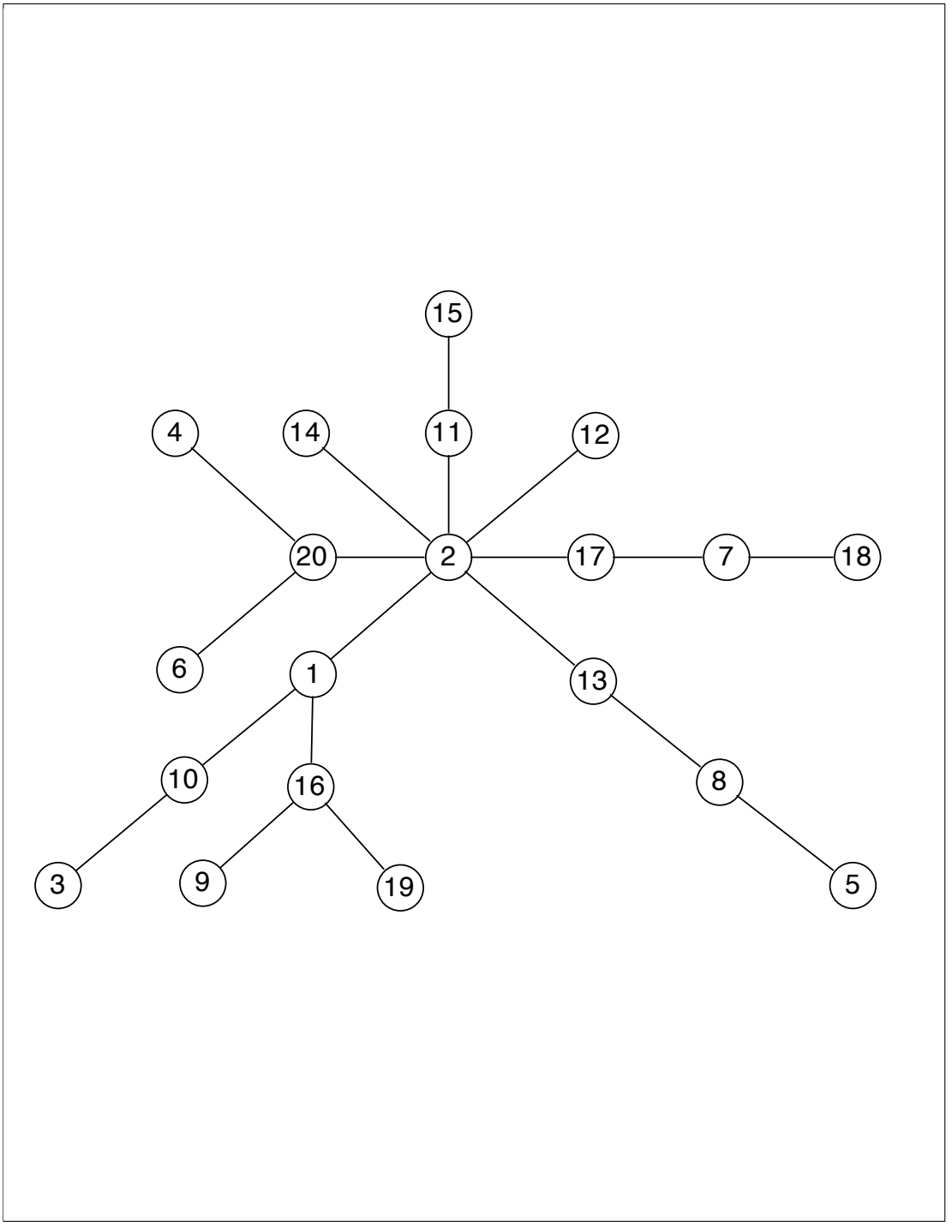
16) plucked; perhaps a cactus

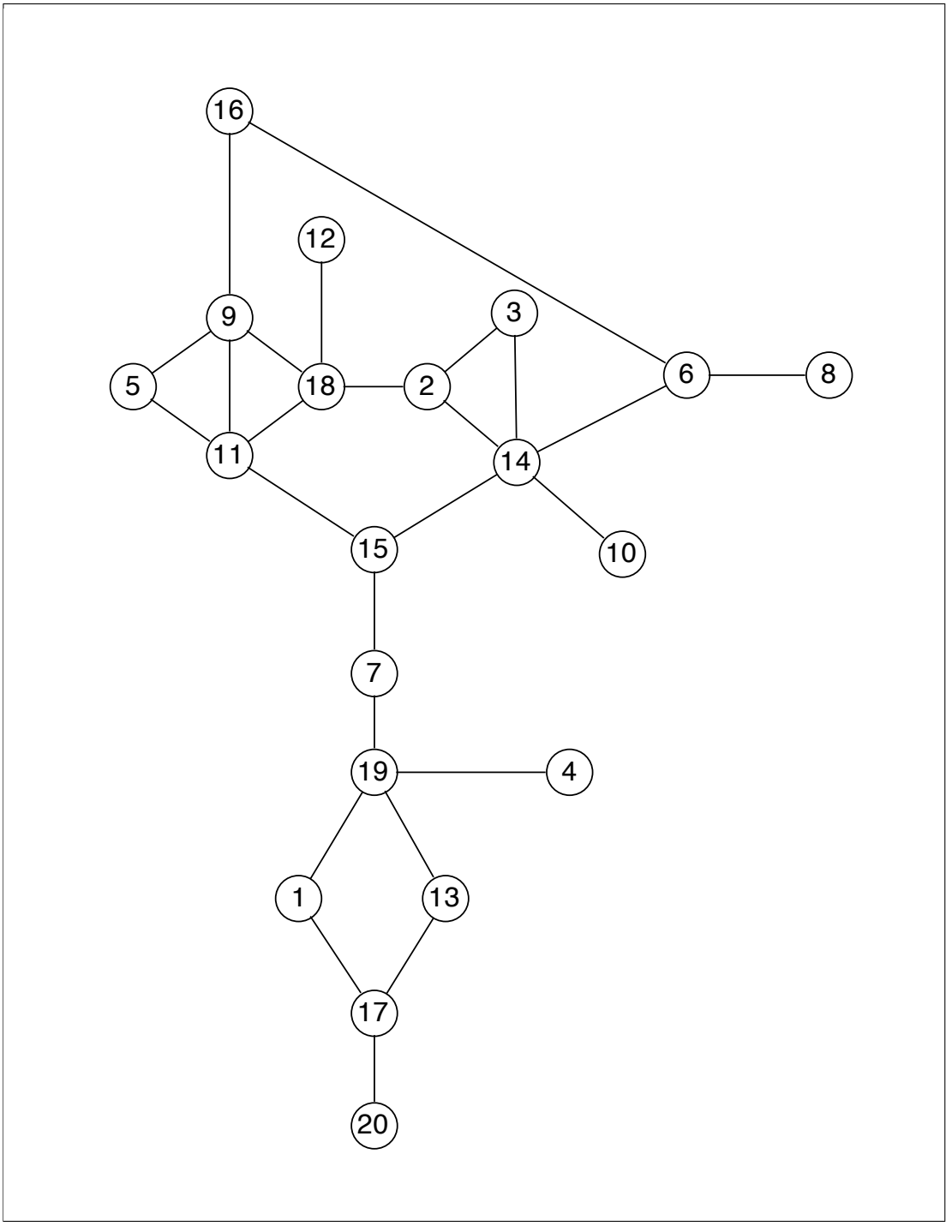
17) bowed

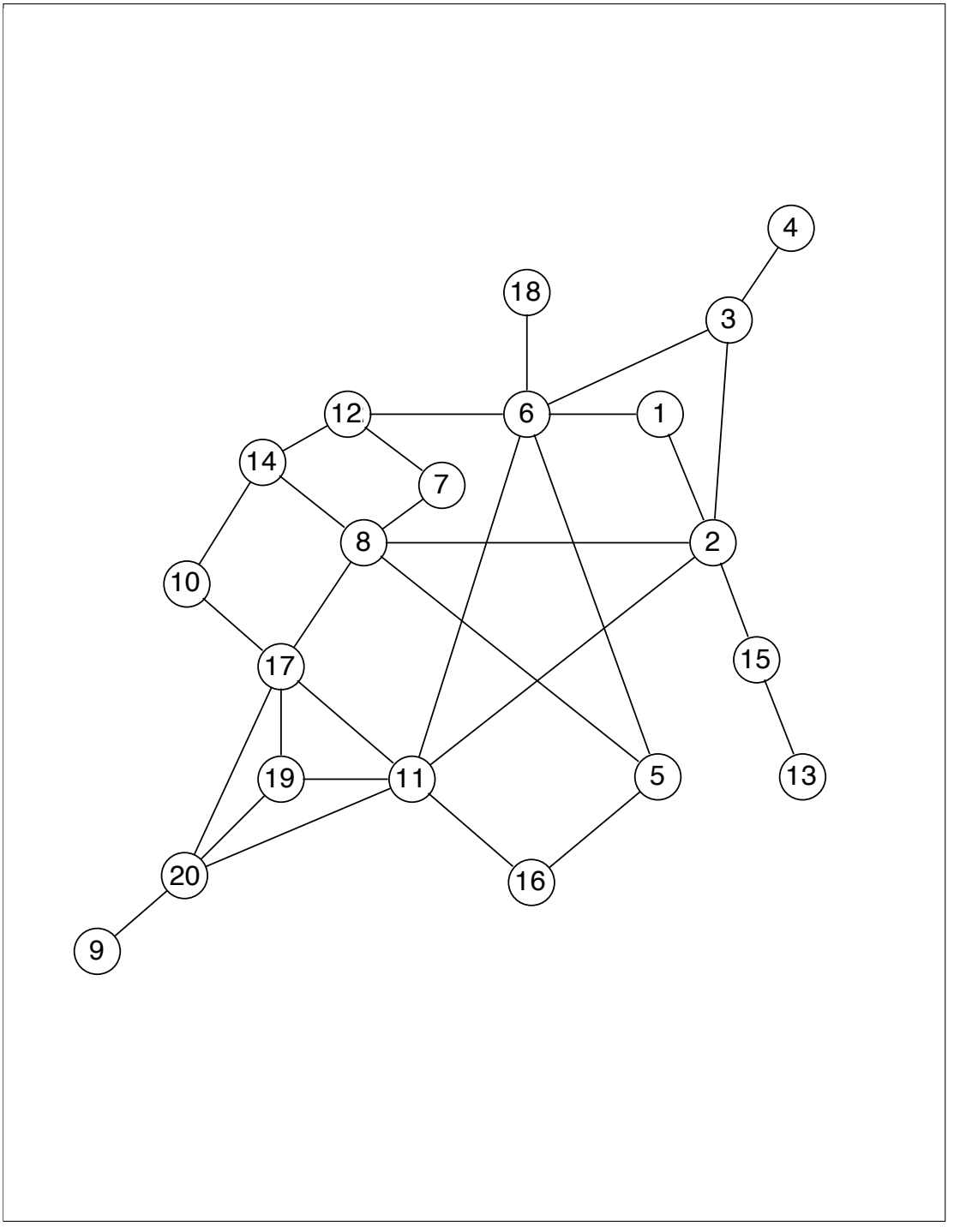
18) a bell-like tone with relatively long decay; allowed to vibrate; followed by silence; possibly repeated several times with the same duration between the attack of each tone

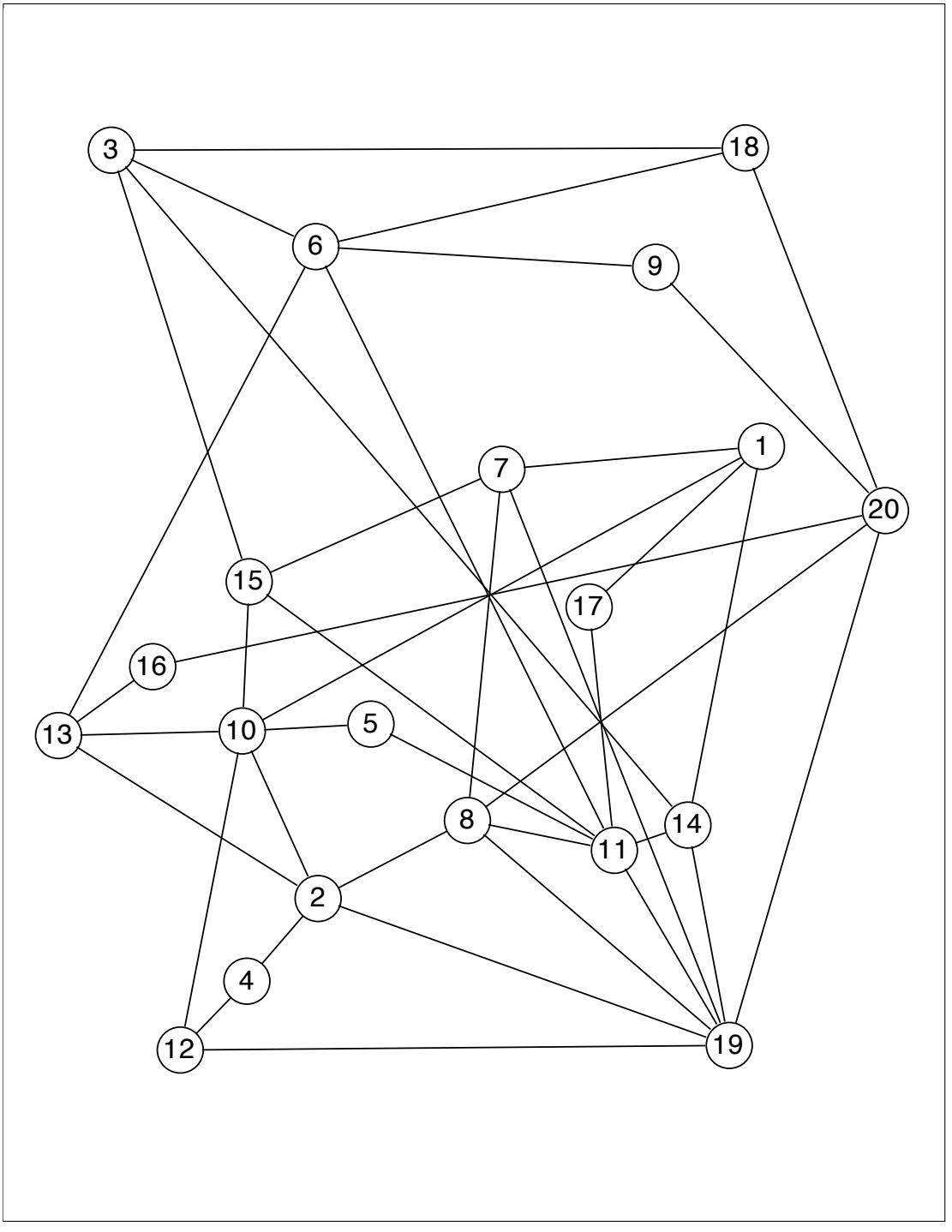
19) silence followed by a repetition or mimicking of no less than 3, nor greater than 8, successive events of another voice

20) silence followed by a repetition or mimicking, irrespective of order, of no less than 3, nor greater than 8, of the events that have just occurred within or without the above listed events


15
4
14
11
12
20
2
17
7
18
6
1
13
10
16
8
3
9
19
5


16
12
9
3
5
18
2
6
8
11
14
15
10
7
19
4
1
13
17
20


4
18
3
12
6
1
14
7
8
2
10
17
15
19
11
5
13
20
16
9


3
18
6
9
1
7
20
15
17
16
13
10
5
8
14
11
2
4
12
19

A.5 *towards completeness*

towards completeness explicitly implements a structural mutation. As explained in the beginning of the appendix, *towards completeness* is the first piece where the mathematical object (which is the mutating structure) is meant to be an ephemeral occurrence with respect to the entire piece, which consist primarily of a long, high-pitched sustained tone.

The elements of the structure are 13 pitches. At first, the graph only connects two elements with one edge. The structure is mutated by adding an edge until the graph is complete; all vertices connect with all others. An edge can only be added if the mutant graph maintains only one connected component. Throughout the transformation, performers take walks on the mutating graph with the primary instruction that only adjacent elements must succeed each other (similar to *small world*). There are additional morphological requirements explained in the performance instructions of the score. Note that there are many possible manifestations of the transformation (towards completeness) of which only one is included below.

towards completeness

for adam overton and quatuor bozzini

the first steps to create a version of the score are to print pages 1 through 78 on transparencies (if not already provided so) and follow this procedure:

- randomly choose one of the transparencies. each one corresponds to an edge in a graph.
- randomly choose from the list of next possible edges/pages. the choices will be visible when placed on top of white opaque paper and correspond to the number on the top left of the transparencies.
- repeat step 2 till all the transparencies have been chosen. when stacked on top of each other, the list of next possible edges/pages will change accordingly.

  note: each player does the above procedure independently. it is advised to keep the pages stacked in order with the most recently chosen edge/page on bottom. it may also be helpful to keep an ordered list of the chosen edges/pages. a custom written computer program is available that automatically generates a score.

a realization is composed of 16 sections diagrammed on the following page, which contains additional performance instructions as well as further instructions for creating a version of the score. the first and last sections consist exclusively of silence. the second and second to last sections consist exclusively of a sustained high-pitched tone, which sustains through the other middle sections. this tone should be as continuous/unchanging as possible. its entrance and exit may be accented in unison by a pitched-percussion instrument of long decay. for the entrance accent, the percussion instrument is allowed to let vibrate. the exit accent is punctuated (short).

in the other sections, the performers not sounding the high sustained tone interpret the graph by exploring morphologies (or phrases/paths) in the structure as follows:

- successive pitches in a morphology must be connected by edges in the graph.
- morphologies are at least 2 tones long and at most the number indicated in the section diagram.
- pitches do not repeat within a morphology. however, the first pitch of a new morphology may be the same as the last pitch of the previous morphology. phrases should generally ascend or descend within the pitch-space.
- except for the ending tone of a morphology, all others are of equal duration. this duration may (and should) vary from morphology to morphology from as much as 2 seconds per tone to the minimum tone duration given in the section diagram.
- except for the ending tone of a morphology, all others are either slurred or punctuated. slurred morphologies should occur more often than punctuated ones.
- typically, the ending tone of a morphology may be punctuated or held any duration less than the approximated morphology length up until the ending tone. However, the last tone should occasionally sound for a *very* long time. punctuated ending tones should occur with equal frequency to longer ending tones.
- a morphology may be followed by silence of any duration less than the approximated morphology length.

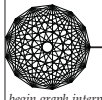
  note: within the limits set forth in the instructions, performers should explore various possibilities to the extent possible. For example, by varying paths through the graph and morphology lengths.

* for all performers an equal dynamic. throughout, clear but not loud.

mike winter (2008)

towards completeness

section number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
section duration	≥ 1′	≥ 5′	18″ / 36″	17″ / 34″	16″ / 32″	15″ / 30″	14″ / 28″	13″ / 26″	12″ / 24″	11″ / 22″	10″ / 20″	9″ / 18″	8″ / 16″	7″ / 14″	≥ 5′	≥ 1′
events	silence	*initiate sustained high-pitched tone*	0″ 18″ 35″ 51″ 1′06″ 1′20″ 1′33″ 1′45″ 1′56″ 2′06″ 2′15″ 2′23″ 2′30″												*end sustained high-pitched tone*	silence

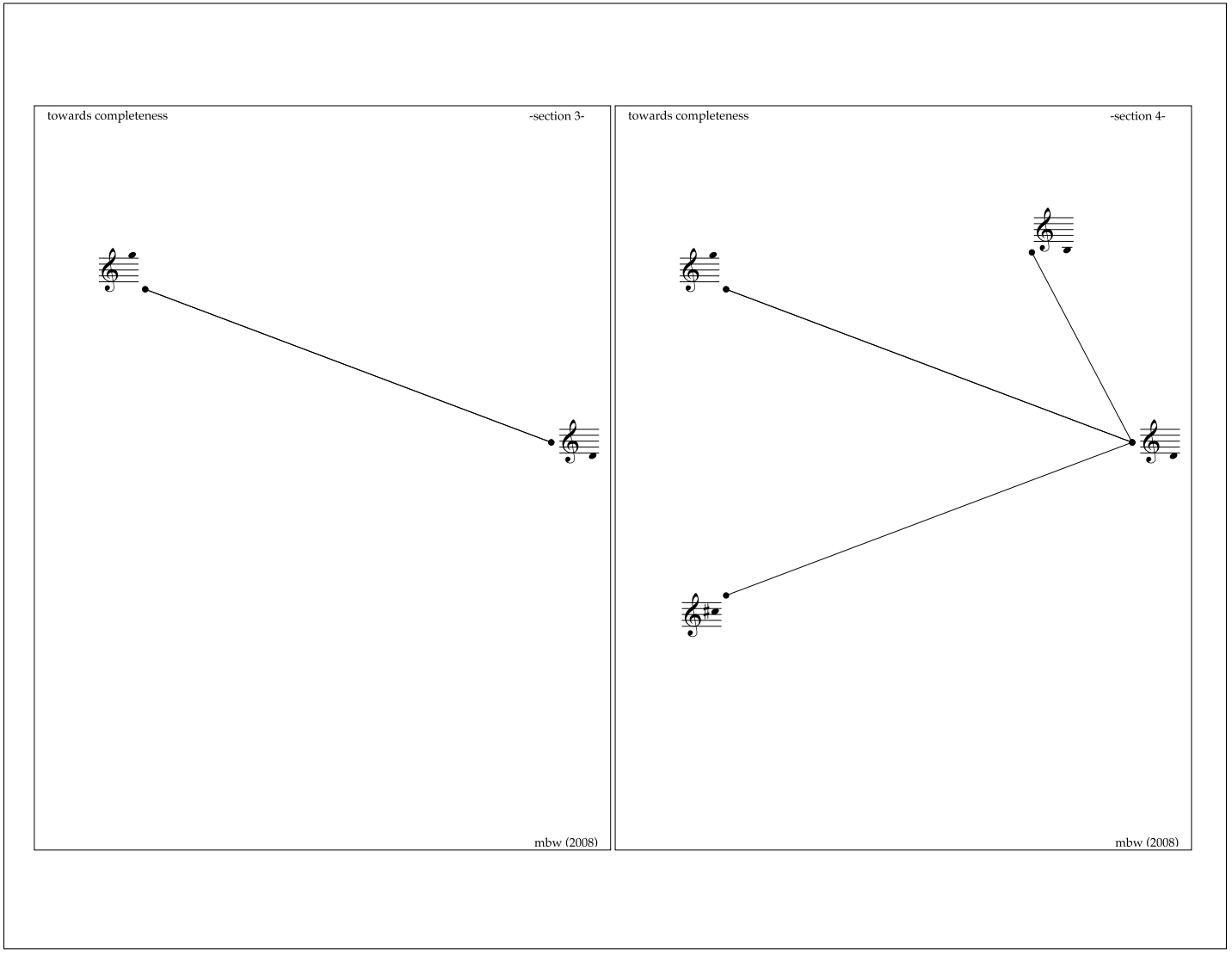


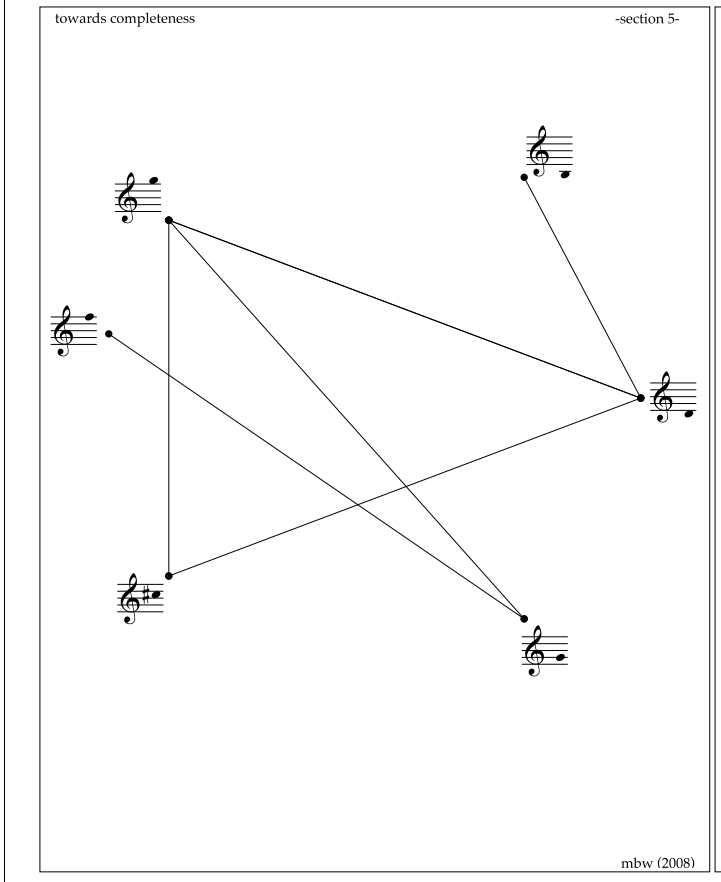
begin graph interpretation — *end graph interpretation*

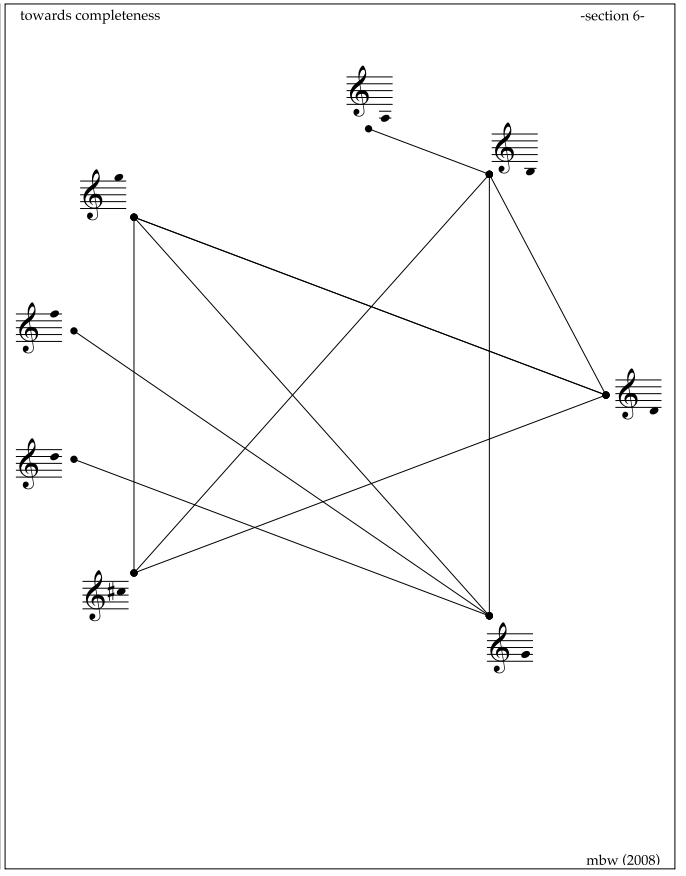
minimum tone durarion	1″	.74″	.58″	.46″	.37″	.29″	.23″	.17″	.12″	.08″	.04″	0″
maximum morphology length	2	3	4	5	6	7	8	9	10	11	12	13
number of edges added	1	2	3	4	5	6	7	8	9	10	11	12

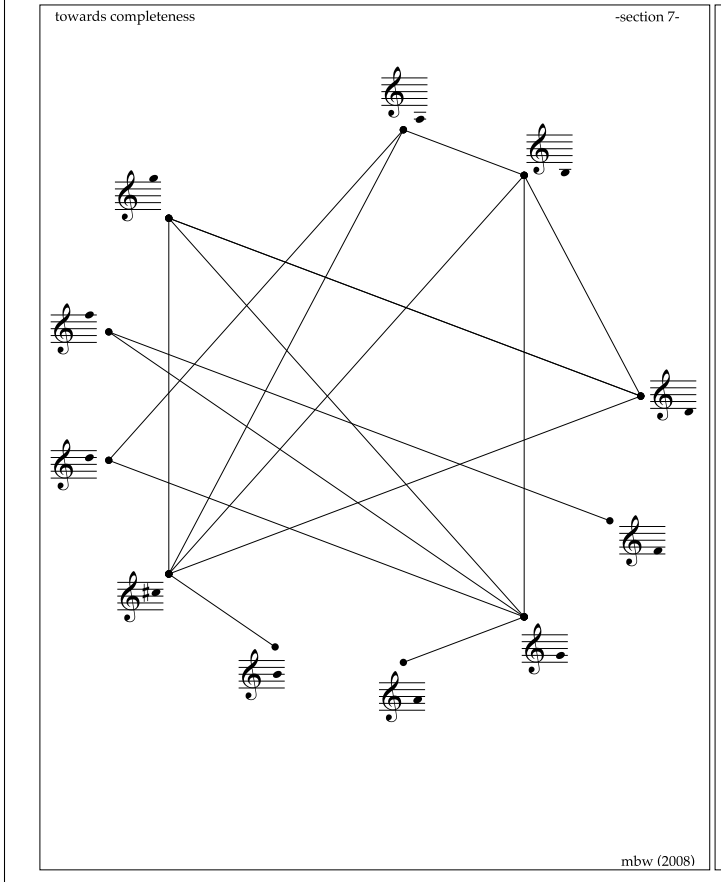
• sections 1 and 16 should be the same the same duration as well as sections 2 and 15, which should be longer. both, preferably long. only sections 3 through 14 are spaced proportionally to time in the diagram.
• the durations of sections 3 through 14 may be scaled uniformly by any factor such that their total duration is less than or equal to half the duration of section 2. a scaling by a multiple of 2 is given. time markings starting from section 3 are given for the unscaled duration.
• the high-pitched sustained tone starting at the beginning of section 2 should be the highest pitch in the piece and sustain continuously till the end of section 15. perhaps a harmonic.
• performers interpreting the graph may enter staggered; however, at the end of section 14, they must stop simultaneously with a punctuated tone.
• the number of edges added is the number of transparencies added in the order determined by the procedure on the preceding page, which ensures a connected graph. by section 14, the graph is complete in that all vertices will be connected to all other vertices. it is perhaps best to make copies (on white paper) beforehand so that their is a page for each section; each with the proper number of edges—1, 3, 6, 10, 15, 21, 28, 36, 45, 55, 66, 78 for sections 3 through 14. versions of the score generated by custom computer software are provided as examples as well as usable versions. the software can be obtained to generate new versions instead of using the transparencies.
• the following tuning across all octaves is preferred over 12-tone equal temperaments (written as the nearest pitch in 12-tone equal temperament with a deviation in cents)—G+0, A+4, B-14, C#-49, D+2, F-31. another option is to predetermine an arbitrary cent deviation up to or down to 50 cents from each written pitch in 12-tone equal temperament. the deviations should be the same for all performers.

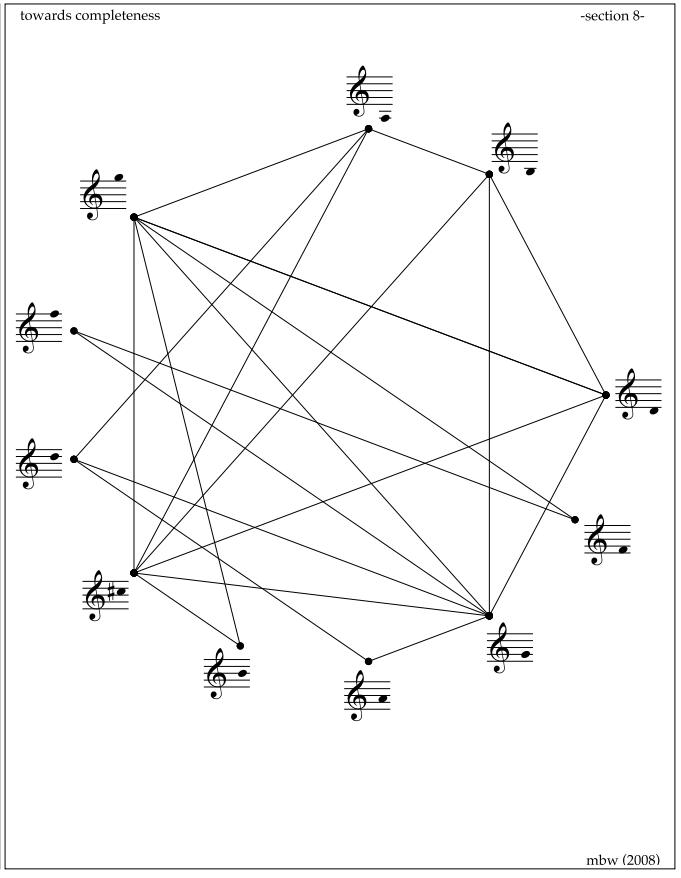
mbw (2008)

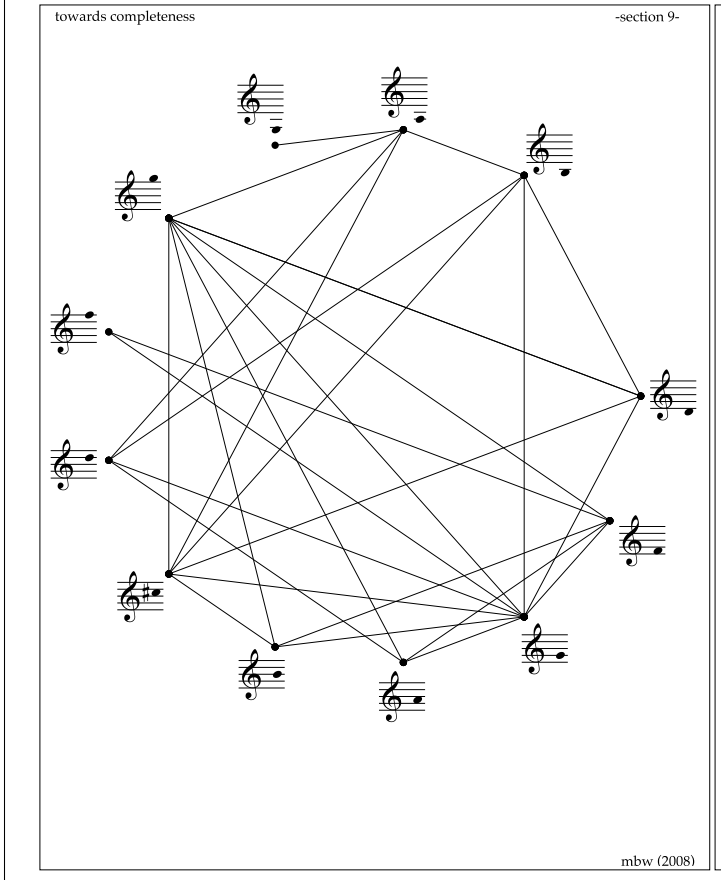

towards completeness
-section 3-
mbw (2008)
towards completeness
-section 4-
mbw (2008)

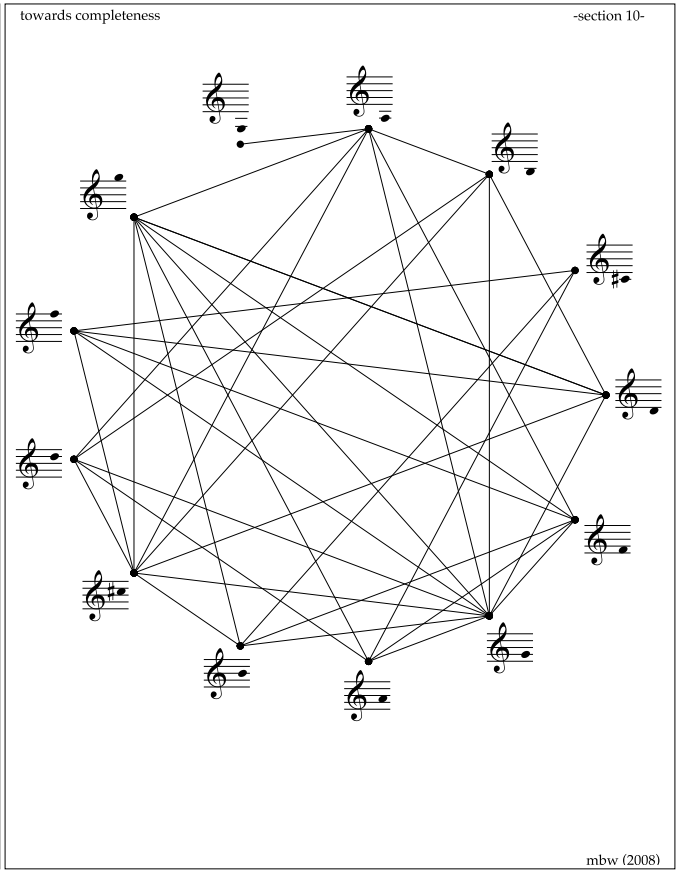

towards completeness
-section 5-
mbw (2008)

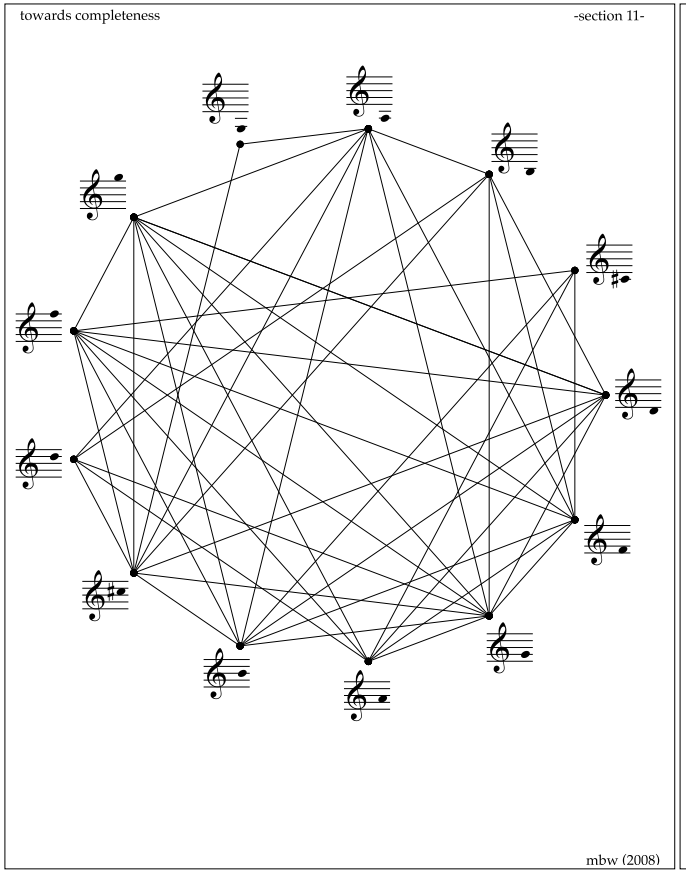

towards completeness
-section 6-
mbw (2008)

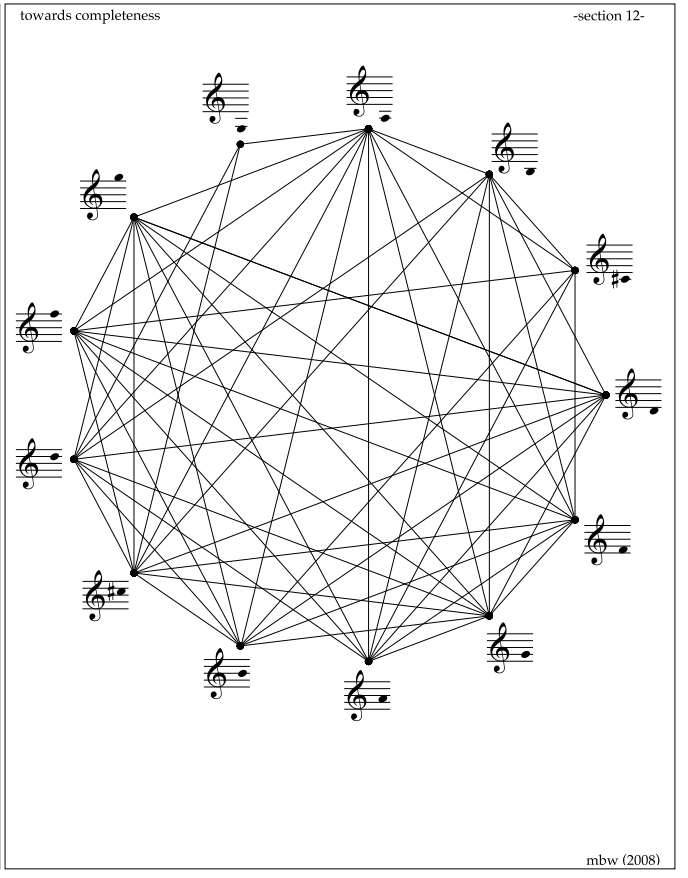

towards completeness
-section 7-
mbw (2008)

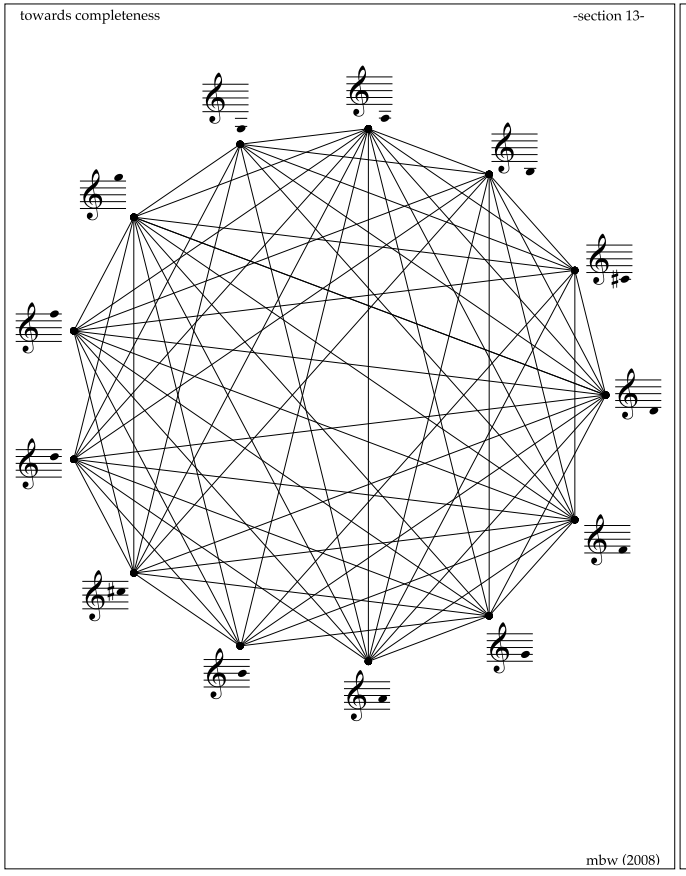

towards completeness
-section 8-
mbw (2008)

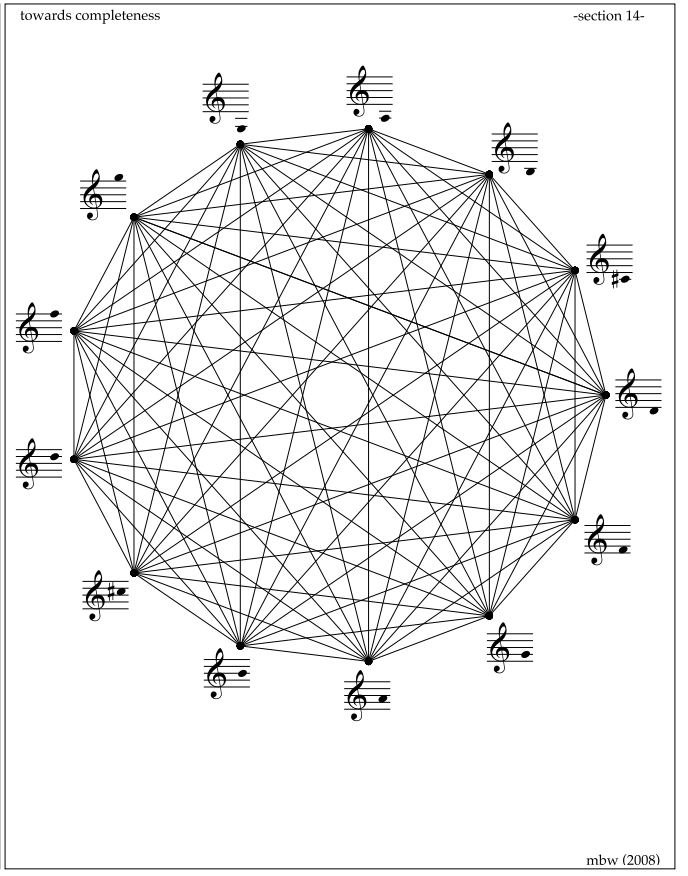

towards completeness
-section 9-
mbw (2008)


towards completeness
-section 10-
mbw (2008)


towards completeness
-section 11-
mbw (2008)


towards completeness
-section 12-
mbw (2008)


towards completeness
-section 13-
mbw (2008)


towards completeness
-section 14-
mbw (2008)

A.6 *for Sol Lewitt*

for Sol Lewitt is an attempt at defining a work of music primarily by relativities such that the piece is scalable to the extent possible (as in many of Sol LeWitt's works—hence, the namesake/dedication). Like *small world*, it is also a study of structural isomorphism from realization to realization. Just as LeWitt's pieces are defined relative to the horizontal and vertical sizes of a given wall, *for Sol LeWitt* is defined relative to an arbitrary pitch range and duration. Another goal was to define the piece in as few words as possible such that there is no redundant information. This hints at a deeper level of structure provided by a description that strives to be elegant in the algorithmic information sense. While the algebra and geometry of the piece are perhaps best communicated by text alone and imagined in mind, Fig. A.3 and A.4 provide visual analogs of the piece where the x-axis is time and the y-axis is pitch.

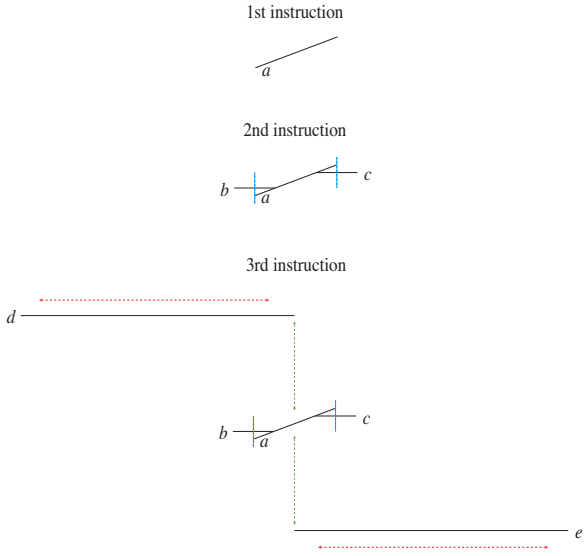


Figure A.3: *for Sol LeWitt*: illustration of instructions

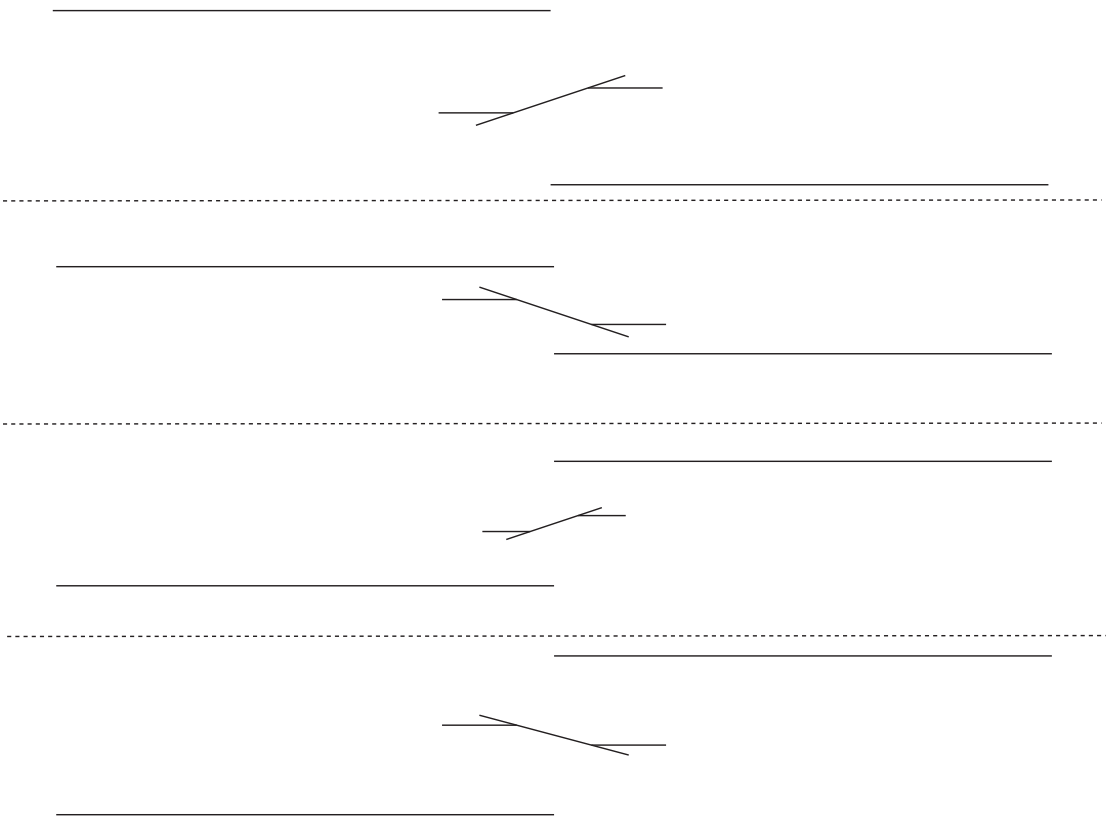


Figure A.4: *for Sol LeWitt*: 4 visual realizations

for Sol LeWitt

1 glissando; 4 sustained tones

a glissando:

- that has a minimal, almost imperceptible slope.
- that starts and ends at the midpoints in time of two sustained tones with the same duration (at most half the duration of the glissando); the first of which ends and the second of which begins in unison with the glissando.
- with a midpoint in pitch that is equidistant to two (other) sustained tones with the same duration (at least three times the duration of the glissando); the first of which ends and the second of which starts at the midpoint in time of the glissando.

the entrances of the sustained tones are preferably accented with percussive attacks that decay slowly and the exits are preferably accented with percussive attacks that are punctuated.

long; clear; not loud.

–michael winter
(february, march, april 2009;
new york city, san francisco, los angeles)

A.7 *for gregory chaitin*

for gregory chaitin plays on the notion of incomputability; of lack of structure. This piece contains a realization of a subset of the halting probability's binary expansion (first defined by the namesake of the piece, Gregory Chaitin [27]). The halting probability is the the probability that a program generated by a toss of a fair coin will halt for some universal prefix-free turing machine. To know each bit n in the expansion, one must know whether or not all computer programs of length n halt. Because this is incomputable, the halting probability is maximally unknowable; has no structure. However, the first 64 bits of the halting probability for a particular universal prefix-free turing machine have been shown by Cristian Calude et al. [22].

The total lack of structure of the halting probability is predicated on an infinite expansion. In practice, it is unclear what it means to embed a finite subset of an unbounded, maximally complex object. Like *towards completeness*, most of the piece consist primarily of a long, high-pitched sustained tone in which the realization of part of the halting probability is a very brief occurrence somewhere in the middle. This punctuation is thus more than just ephemera; it is a juxtaposition of opposing logics to the most extreme degree. An extremely simple object exists (the high-pitched sustained tone) with an extremely complex one (the realization of the halting probability).

The score details how to realize the binary expansion of the halting probability.

for gregory chaitin

• a continuous tone with a distinct pitch sustained for a long time. the entrance of this tone is preferably accented with a percussive attack that decays slowly. the exit is preferably accented with a percussive attack that is punctuated.

• a realization of a subset of at least 8 bits (preferably more) of the halting probability for some universal prefix-free turing machine.* the sequence of digits are read linearly, '0' and '1' representing two distinct events, which remain constant throughout. each event should create a sound with partials that interact/interfere with the sustained tone.

• the piece starts and ends with the sustained tone alone for a duration at the very least twice the length of the realization of the halting probability, which is ephemeral with respect to the total duration of the piece.

• perhaps with the performers and instruments out of view or in a dark space with the performers and/or instruments dimly (and directly) lit if necessary. perhaps as an installation. allowed to repeat. possibly with silence between repeats.

• clear. not loud. delicate. almost still. concerned with the phenomena of sound itself.

* an example of a halting probability is provided below. see anything written by chaitin on the number 'omega' (or chaitin's constant), which is the halting probability for some universal prefix-free turing machine. the given bit string was calculated by cristian calude et al. in 'computing a glimpse of randomness'

0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 0 1 0 0 0 1 1 1 1 1 1 1 0 0 1 0 1 1 1 0 1 1 1 0 1 0 0 0 0 1 0 0 0 0

-michael winter (new smyrna beach, fl; may, 2009)

---

a realization for philip thomas

• grand piano

• the sustained tone should be a note on the piano with three strings. the entrance of the sustained tone is first actuated (and slightly accented) by the hammer and sustained by an e-bow on the middle string. the exit of the sustained tone is slightly accented by a punctuated (very short) strike of the hammer.

• the bits of the halting probability are realized on a note with two strings that is three octaves below the sustained tone. a '0' is realized by striking the strings gently (by depressing the key) while touching a node such that a harmonic of the open string sounds. the particular node may vary from tone to tone ('0' to '0'). perhaps explore random nodes that produce very high harmonics. a '1' is realized by gently striking the open string. all tones allowed to decay to nothing.

A.8 *field and perfect circuit*

The mathematical object embedded in *field and perfect circuit* is an example of a labeled graph used for generative purposes. It is a circuit representation of a perfect squared square, which is a dissection of a square into several squares of distinct area. These types of circuits, called Smith diagrams, have the property that several paths from the positive to negative end of the circuit share vertices and the accumulated sum of the edge lengths from any of the poles to a shared vertex is always equal across different paths. The score includes the Smith diagram of the perfect squared square of lowest order [45]. (For more on squared squares and Smith diagrams see [2].)

In *field and perfect circuit*, the edge labels represent the amount of time that a particular event may occur. All the performer start at the same time and each individually chooses a path from the positive pole to the negative pole playing successive distinct events for a length proportional to the label of each edge that is crossed. The conglomerate timbre shifts in various ways based on how many shared vertices are in the realized paths. Sometimes one or more events start and/or stop at the same time creating more obvious shifts of the conglomerate timbre. At other times, performers change events alone and the conglomerate timbre shifts more subtly.

field and perfect circuit

- a continuous sound sustained for a long time. the entrance of this sound is preferably accented with a percussive attack that decays slowly. the exit is preferably accented with a percussive attack that is punctuated.

- at least 5, preferably more, simultaneous realizations of the smith diagram of some simple perfect squared square or rectangle* (which is essentially a circuit represented by a directed graph) according to the following guidelines and possibilities:
    - the sum of the edge labels in a path from the positive pole to the negative one corresponds to some predetermined amount of time, which is ephemeral with respect to the total duration of the piece. (all paths have the same sum.)
    - the circuit/graph is realized by starting at the positive pole and moving across edges to adjacent vertices until the negative pole is reached.
    - for each edge crossing, an event occurs for the duration corresponding to the edge label (proportional to the length of any path from pole to pole). the start of the first event is accented with a percussive attack that decays slowly. the end of the last event is accented with a percussive attack that is punctuated. the start/end of all other events may be accented with a percussive attack that decays slowly or is punctuated. possible events are provided below listed from top to bottom in order of the amount they should occur. the top two are mandatory; the bottom two optional. the same event cannot repeat successively.
        - a (quasi-)continuous sound with or without a distinct pitch or set of pitches. (favored for longer events.)
        - a short percussive sound quickly repeated at an individual tempo with respect to any other quickly repeated sounds that may already be occurring. (favored for shorter events.)
        - a reading of a text relevant to the piece that is not the text of this score.
        - an event perceived as a single gestalt over the duration indicated by the edge label.
    - if 2 or more events start or end at the same time, the accented attacks must be simultaneous. that is, timing should be executed with precision.
    - preferably, all the realizations together traverse every edge in the diagram at least once. if resources limit the number of paths that can be traversed, then subsets of paths that have several corresponding starts and stops while crossing as many edges as possible should be preferred. not too dense, not too sparse.

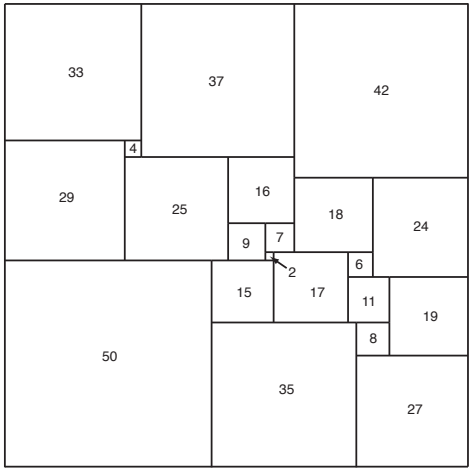
- the piece starts and ends with the long sustained sound alone for a duration at the very least twice the length of the realization of the simple perfect squared square or rectangle.

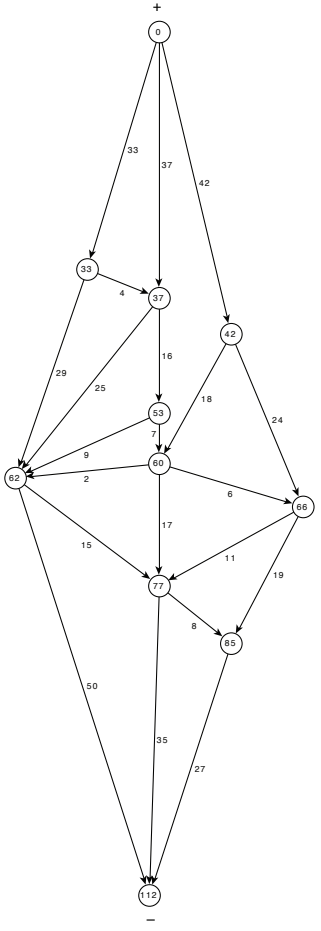
- perhaps with the performers and instruments out of view or in a dark space with the performers and/or instruments dimly (and directly) lit if necessary. perhaps as an installation. allowed to repeat. possibly with silence between repeats.

- clear. not loud. cohesive. primarily concerned with the phenomena of sound itself.

* an example of a perfect squared square, its corresponding smith diagram and possible paths through it are given on the following page. see http://www.squaring.net (accessed may, 2009), which has examples of several perfect squared squares and rectangles, descriptions of smith diagrams and related information.

- *michael winter* (new smyrna beach, fl; may, 2009)





- left: a perfect squared square of lowest order (least number of squares). see a. j. duijvestijn (1978) "simple perfect squared square of lowest order" in the *journal of combinatorial theory*.
- middle: the smith diagram for the duijvestijn perfect squared square.
- right: a list of all possible paths (both by edges and vertices). paths by vertices are most likely easier to read as they give absolute values. for example, if the units of the rectangle are determined to be equivalent to seconds, then the realization of the smith diagram is 112 seconds in total.

paths by edges:	paths by vertices:
{42, 24, 19, 27}	{0, 42, 66, 85, 112}
{42, 24, 11, 8, 27}	{0, 42, 66, 77, 85, 112}
{42, 24, 11, 35}	{0, 42, 66, 77, 112}
{42, 18, 6, 19, 27}	{0, 42, 60, 66, 85, 112}
{42, 18, 6, 11, 8, 27}	{0, 42, 60, 66, 77, 85, 112}
{42, 18, 6, 11, 35}	{0, 42, 60, 66, 77, 112}
{42, 18, 17, 8, 27}	{0, 42, 60, 77, 85, 112}
{42, 18, 17, 35}	{0, 42, 60, 77, 112}
{42, 18, 2, 15, 8, 27}	{0, 42, 60, 62, 77, 85, 112}
{42, 18, 2, 15, 35}	{0, 42, 60, 62, 77, 112}
{42, 18, 2, 50}	{0, 42, 60, 62, 112}
{37, 16, 7, 6, 19, 27}	{0, 37, 53, 60, 66, 85, 112}
{37, 16, 7, 6, 11, 8, 27}	{0, 37, 53, 60, 66, 77, 85, 112}
{37, 16, 7, 6, 11, 35}	{0, 37, 53, 60, 66, 77, 112}
{37, 16, 7, 17, 8, 27}	{0, 37, 53, 60, 77, 85, 112}
{37, 16, 7, 17, 35}	{0, 37, 53, 60, 77, 112}
{37, 16, 7, 2, 15, 8, 27}	{0, 37, 53, 60, 62, 77, 85, 112}
{37, 16, 7, 2, 15, 35}	{0, 37, 53, 60, 62, 77, 112}
{37, 16, 7, 2, 50}	{0, 37, 53, 60, 62, 112}
{37, 16, 9, 15, 8, 27}	{0, 37, 53, 62, 77, 85, 112}
{37, 16, 9, 15, 35}	{0, 37, 53, 62, 77, 112}
{37, 16, 9, 35}	{0, 37, 53, 62, 112}
{37, 25, 15, 8, 27}	{0, 37, 62, 77, 85, 112}
{37, 25, 15, 35}	{0, 37, 62, 77, 112}
{37, 25, 50}	{0, 37, 62, 112}
{33, 4, 16, 7, 6, 19, 27}	{0, 33, 37, 53, 60, 66, 85, 112}
{33, 4, 16, 7, 6, 11, 8, 27}	{0, 33, 37, 53, 60, 66, 77, 85, 112}
{33, 4, 16, 7, 6, 11, 35}	{0, 33, 37, 53, 60, 66, 77, 112}
{33, 4, 16, 7, 17, 8, 27}	{0, 33, 37, 53, 60, 77, 85, 112}
{33, 4, 16, 7, 17, 35}	{0, 33, 37, 53, 60, 77, 112}
{33, 4, 16, 7, 2, 15, 8, 27}	{0, 33, 37, 53, 60, 62, 77, 85, 112}
{33, 4, 16, 7, 2, 15, 35}	{0, 33, 37, 53, 60, 62, 77, 112}
{33, 4, 16, 7, 2, 50}	{0, 33, 37, 53, 60, 62, 112}
{33, 4, 16, 9, 15, 8, 27}	{0, 33, 37, 53, 62, 77, 85, 112}
{33, 4, 16, 9, 15, 35}	{0, 33, 37, 53, 62, 77, 112}
{33, 4, 16, 9, 35}	{0, 33, 37, 53, 62, 112}
{33, 4, 25, 15, 8, 27}	{0, 33, 37, 62, 77, 85, 112}
{33, 4, 25, 15, 35}	{0, 33, 37, 62, 77, 112}
{33, 4, 25, 50}	{0, 33, 37, 62, 112}
{33, 29, 15, 8, 27}	{0, 33, 62, 77, 85, 112}
{33, 29, 15, 35}	{0, 33, 62, 77, 112}
{33, 29, 50}	{0, 33, 62, 112}

A.9 *recitation, code, and (perhaps) round*

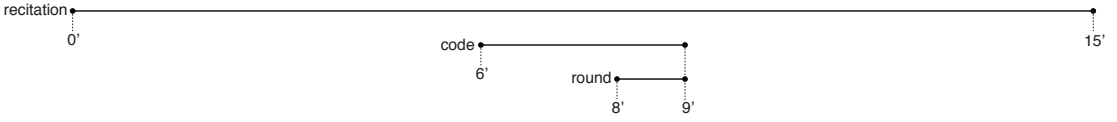
recitation, code, and (perhaps) round consists of two or three parts: a recitation (plain; chant-like), a code (rising from the ether; cacophonous), and (perhaps) a round (glorious; strident) occurring at overlapping times. The piece sets a poem entitled "Now Trips a Lady, Now Struts A Lord" by Elizabeth Winder. The poems contains 16 words. The code part of the piece incorporates coding theory (hence the name). Performers are instructed to sing each word and realize with a percussion instrument a unique binary code associated with each word (detailed in the score). Each binary word contains four bits. Since there are 16 words, the codewords are equivalent to a Huffman code [59] with no compression.

The round consists of a ground melody that may be altered by the performers such that the alteration is similar to the ground. This produces similar results to the more rigorous structural mutations referred to in the thesis.

recitation, code, and (perhaps) round

fifteen minutes for choir with metallic or wooden percussion instruments
text: Now Trips a Lady, Now Struts a Lord; *Elizabeth Winder (2009)*
written for the Canticum Ostrava for premiere at the 2009 Ostrava Days Festival
-michael winter (la, 2009)

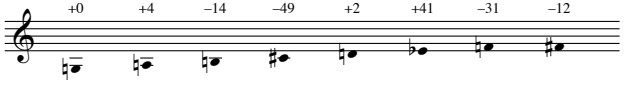
• two or three parts: a recitation (plain; chant-like), a code (rising from the ether; cacophonous), and (perhaps) a round (glorious; strident) occurring at overlapping times (as diagrammed below). each individual part is described separately on the following pages. if possible, performers should distribute themselves throughout the performance space. singing should be minimal with no vibrato or other embellishments.



• the code and the round should end abruptly. all players end his or her currently sung tone simultaneously with the others.

• the round is optional and can be sung independently if the occasion warrants.

• note that the pitches of the piece derive from a harmonic series. in the more traditional music notation provided in the score, above each note is a corresponding cents-deviation (one-hundredth of a tempered semitone) from the nearest pitch in twelve-tone equal temperament. performers should familiarize themselves with the scale below transposed into his or her respective range.



Now Trips a Lady, Now Struts a Lord

Lost my cat-

gland in the Chat

Sanctus

[sans-culotte]

blot

from view, My Wick

You--

• recitation (bass)

  • a repeated recitation of the poem on a low g. slow. with drawn-out vowels and variable pacing of each syllable. repeated for the entire duration of the piece, which may only end on the final word. clear. not loud.

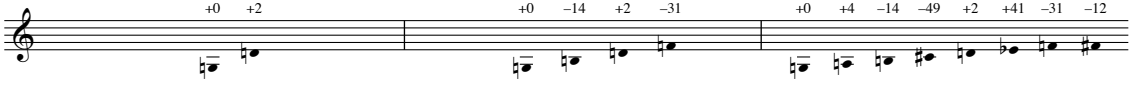
    • this part may be solo or with several low voices. in the latter case, the ensemble should experiment with homophonic recitations (with the start and end of each syllable cued by one performer at a time). the ensemble may also try independent recitations per performer. in this case, the performers should enter and exit staggered with the first and last recitation sung by only one performer.

    • the recitation may be accompanied by a continuous, pitched tone in unison with and sustained throughout the recitation. the entrance of the recitation and possible accompaniment may be accented with a percussive attack that decays slowly. the exit may be accented with a percussive attack that is punctuated. if the recitation is accompanied, the performer shall feel free to take longer pauses between words and repetitions of the poem.

• code (as many vocalists as possible—in all ranges)

  • each performer sings words from the poem chosen in succession arbitrarily (that is, no particular order). pitches are also chosen arbitrarily from a set. the first, second, and third measures in the music notation below give the sets for time 6' – 7', 7' – 8' and 8' – 9', respectively. pitches may be transposed by any number of octaves. one pitch per word. the general texture goes from sparse to dense. to ensure this trajectory, durations of the tones should start long and get shorter (and louder) throughout the three minutes of the part.

  • available pitches



  • during the singing of each word, the performer realizes, with their respective percussion instrument, the code associated with the given word (provided below; read left to right) as follows: disturbing the space to the extent possible, 'zeroes' are realized as one hard strike of the instrument and 'ones' are realized as loud rattles or rolls.

    • for example, if the player plays a metal pot, a 'zero' could be realized as a hard strike on the outside, while a 'one' could be realized as a vigorous shaking of the beater inside the pot that strikes the (in)sides in rapid (and perhaps arhythmic) succession.

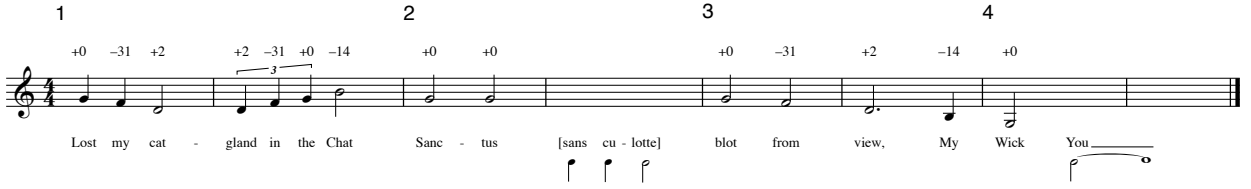
lost	my	cat	gland	in	the	chat	sanctus
0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1
sans	culotte	blot	from	view	my	wick	you
1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1

• round (four groups—each in any range)

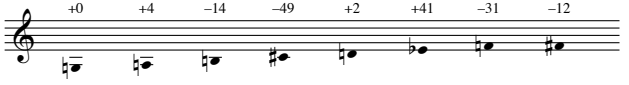
• performance of the round should be at a rather fast tempo and should sound at a dynamic that matches or is slightly above the rest of the ensemble. the round is composed of two possible parts: a ground and possible alterations. the first two groups of performers must start with the ground. successive groups may start with the ground or an alteration. the numbers above the ground indicate the start time of each group relative to the start of the round. after one reading of the poem performers are free to alternate between the ground and alterations.

• ground

• the ground of the round (shown below in traditional notation) may be transposed by any number of octaves. where durations are given below the words (with no notes in the staff above), performers are allowed to choose from any of the available pitches (again in any octave) notated below the ground.



• available pitches



• alteration

• an alteration is any manipulation of the ground that preserves the order of words in the poem. pitches must be from the above set (including octave equivalents). one pitch per syllable. the preferred method of alteration is constructed as follows. partition the sequence of symbols below such that the partitions *generally* contain more than 3 and less than 7 symbols. then rearrange the partitions creating a new sequence. performers may create several of these sequences prior to performance. an alteration is realized by starting on any pitch and reading the entire sequence of symbols from left to right. ↑, ↓, and — indicate that the next pitch *must be* higher, lower, or the same, respectively. ↕ indicates that the performer *may choose* whether the next pitch is higher, lower, or the same. one pitch per syllable. performers may group any number of words into a phrase. tones should generally be short in duration however the last note of a phrase may sustain longer. performers may try to alter and realize the sequence during performance (that is, in 'real-time') by realizing disjunct subsequences of the sequence below in no particular order.

A.10 *Approximating Omega*

The first section of *Approximating Omega* consists of a speaker with accompaniment explaining a slightly modified version (for artistic purposes) of Chaitin's toy Lisp from his 1994 book, "The Limits of Mathematics" [31]. This lisp consists of single symbols/atoms reserved for primitive functions. Each symbol in the language is mapped to a sound. The second section is a transcription of Chaitin's computer program that approximates Omega, the probability that a random program will halt on a self-delimiting universal Turing machine. The symbols are realized linearly one by one by their respective sounds. In addition, any routines in the program bound by parenthesis also have a continuous sound that starts and ends with the parenthesis that delimit the routine.

The structure of the piece is completely predicated on the structure of the program, which has nested routines 17 levels deep at the maximum depth. This structure is easily seen in the first of the traditionally notated transcriptions of the program. Apart from a structure borrowed from the program itself, the transcription of a computer program that approximates a maximally complex number such as Omega is perhaps the most pure realization of such a mathematical object since any computation or subset (as in the piece, *for Gregory Chaitin*) does not so fully encapsulate the phenomenon.

Approximating Omega

Note (to be used as an accompanying document for performances providing resources related to this piece):

In a 1975 paper entitled “A theory of program size complexity formally identical to information theory,” Gregory Chaitin formally defines Omega, which is the probability that a computer program with bits generated by tosses of a fair coin halts on a universal self-delimiting Turing machine. Omega is maximally complex and incomputable because no algorithm can determine whether an arbitrary computer program halts. The undecidability of what is now known as the “halting problem” was originally shown by Alan Turing in his 1936 paper, “On computable numbers, with an application to the Entscheidungsproblem,” as a corollary to a computer theoretical proof of Kurt Godel’s incompleteness theorem first presented in Godel’s 1931 article, “On formally undecidable propositions of Principia Mathematica and related systems I.”

While Omega is incomputable, Chaitin has written a computer program that approximates Omega with increasing accuracy over longer and longer amounts of time. The program is written in a version of LISP that Chaitin extended and altered from the original LISP protocol first created by James MCarthy 1958 with a published explanation in his 1960 paper, “Recursive functions of symbolic expressions and their computation by machine.”

The first [optional] section of this piece consists of an accompanied speaker reading an explanation of Chaitin’s dialect of LISP. The text is adapted from Chaitin’s 1994 book, “The Limits of Mathematics.” Chaitin has further extended his 1994 LISP in his 2001 book, “Exploring Randomness.” The decision to use Chaitin’s older version of LISP and make changes to the text of “The Limits of Mathematics” are for artistic purposes with permission from the author.

The second section realizes symbol-by-symbol the program that approximates Omega. Distinct (primarily short) sounds represent each symbol and the entrances and exits of various continuous sounds demarcate the expressions in the program. Thus, the nesting of subroutines within the program completely determines the form of the music.

-michael winter (la; 2010)

Performance Instructions:

Dynamics and Setting:
All individual instruments should sound at a uniform dynamic; each sound heard clearly without being loud. Sound sources should be distributed throughout the performance space. Realizations should be primarily concerned with the phenomenon of sound itself.

Instrumentation:
Part 0 comprises a battery of 26 different sounds indicated by number. 0 and 1 indicate pitched sounds (pitch class – D; any fixed octave) from a percussion instrument with long natural decay. 0 indicates to let ring. 1 indicates a punctuated tone (quickly stopped). Sounds 0 and 1 occur most frequently. When possible, distribute these occurrences to a set of a particular instrument (such as 17 chimes of the same pitch located throughout the performance space) such that: when several 0s occur in succession, each sound comes from a different source; when a 0 is followed shortly by a 1, the same source produces both sounds (that is, the tone initially allowed to ring is stopped by the punctuated tone that follows). Each remaining sound-number in Part 0 indicates a distinct, non-pitched sound that is short and/or has a short natural decay. Distribute the sources of these sounds throughout the performance space to the extent possible. Each part from 1 through 17 must be distinguished uniquely by timbre and/or pitch. One performer may play several parts with double-stops, multiple instruments, polyphonic synthesized or recorded sounds, etc.. Assign each part played by a pitched instrument a distinct pitch derived from one of the first seventeen primes of the harmonic series of D. Note names with cent deviations (one hundredth of a tempered semitone) are provided above the charts in Section 2. Performers may sound a part's respective pitch-class in any octave and may change octaves for each new tone.

Section 1 (optional):
The text is read in a paced speaking voice with silences of varying lengths between each paragraph (delimited by double line breaks). Varying subsets of the ensemble accompany each paragraph starting and ending precisely with the speaker. The performers of Part 0 accent the start and end of each paragraph with sounds 0 and 1, respectively. Per paragraph, the accompanying sound should seem as a single gestalt. Sounds should generally by continuous however percussion instruments may choose to sound a series of punctuated attacks repeated quickly at individual tempi. Individual instruments may enter and exit freely so long as an overall continuity remains. Symbols bound by brackets are not spoken. Instead, a tone corresponding to the sound-number of Part 0 (given in the left column) sounds. Precede and succeed this tone with silence. The paragraph is interpreted to start on the following line. A word in quotes coincides with the sounding of a tone corresponding to the sound-number of Part 0 (also given in the left column). Underlined words are neither spoken nor accompanied. Performers of Part 0 must play when indicated by brackets or quotes in the text but may also contribute to the more continuous accompanying sound of each paragraph.

Section 2:
Part 0 is notated as a list of pairs. The first number is a time-unit within which a sound indicated by the second number occurs. Two charts provide options for the remaining 17 parts (organized by columns). The column contains a list of time-unit pairs for the start and end times of each tone. The tones must enter and exist precisely with the sound in Part 0 occurring in the same time-unit. The ensemble may uniformly scale the time-units by any amount. Note that in Option 1, the higher the part number, the shorter the general durations of tones. Option 2 sacrifices this for parts that are more uniformly active. More traditionally notated scores are also provided for both options. An appendix gives the list of time pairs without part assignment. The ensemble may explore distributing the tones in other ways but it will always take at least 17 parts.

Section 1:

Part 0: Speaker with Accompaniment

Omega is formally defined as the probability that a computer program with bits generated by tosses of a fair coin halts on a universal self-delimiting computer. Since no algorithm can determine whether an arbitrary computer program halts, Omega is maximally complex and incomputable. However, there exists a computation that approximates Omega with increasing accuracy over longer and longer amounts of time. We outline this method in an altered and extended version of the computer programming language, LISP.

LISP more closely resembles fundamental subjects such as set theory and logic than a traditional programming language. The LISP formalism consists of several primitive functions and a set of rules for defining more complex functions from the initially given primitives. LISP functions are technically known as partial recursive functions. Data and function definitions in LISP consist of S-expressions. S stands for symbolic.

S-expressions are lists consisting of start and end delimiters binding zero or more elements, which may be atoms or sublists. Formally, the class of S-expressions is the union of the class of atoms and the class of lists.

The fundamental semantic concept of LISP is that of the value of an S-expression in a given environment. An environment consists of an associated list in which variables (atoms) and their values (S-expressions) alternate. If a variable appears several times, only its first value is significant. If a variable does not appear in the environment, then it is a literal constant in that it itself is its value.

LISP reserves the following symbols given with a name, the number of arguments (if applicable) and an explanation. All but the first four represent primitive functions.

0 Symbol: [(]
Name: Start-Delimiter
Explanation: Denotes the start of an S-expression.

1 Symbol: [)]
Name: End-Delimiter
Explanation: Denotes the end of an S-expression.

2 Symbol: [1]
Name: True
Explanation: Denotes the value true.

3 Symbol: [0]
Name: False
Explanation: Denotes the value false.

4 Symbol: [']
Name: Quote or Literally
Arguments: 1
Explanation: The result of applying this function is the unevaluated argument expression.

5 Symbol: [.]
Name: Atom
Arguments: 1
Explanation: The result of applying this function to an argument is true or false depending on whether or not the argument is an atom.

6 Symbol: [=]
Name: Equal
Arguments: 2
Explanation: The result of applying this function is true or false depending on whether or not the two arguments are the same S-expression.

7 Symbol: [+]
Name: Head" or "First
Arguments: 1
Explanation: The result of applying this function to an atom is the atom itself. The result of applying this function to a non-empty list is the first element of the list.

8 Symbol: [-]
Name: Tail" or "Rest
Arguments: 1
Explanation: The result of applying this function to an atom is the atom itself. The result of applying this function to a non-empty list is the remaining elements after deletion of the first element. Thus, the tail of an (n + 1)-element list is an n-element list.

9 Symbol: [*]
Name: Join
Arguments: 2
Explanation: If the second argument is not a list, then the result of applying this function is the first argument. If the second argument is an n-element list, then the result of applying this function is the (n + 1)-element list whose head is the first argument and whose tail is the second argument.

10 Symbol: [,]
Name: Display
Arguments: 1
Explanation: The result of applying this function is its argument and is used to display intermediate results. In other words, this is an identity function. It is the only primitive function with a side-effect, which is to display the argument.

11 Symbol: [/]
Name: If-then-else
Arguments: 3
Explanation: If the first argument is not false, then the result is the second argument. If the first argument is false, then the result is the third argument. The argument that is not selected is not evaluated.

12 Symbol: [!]
Name: Evaluate
Arguments: 1
Explanation: The expression that is the value of the argument is evaluated in an empty environment. This is the only primitive function that is a partial rather than a total function.

13 Symbol: [?]
Name: Try or Depth-Limited Evaluation
Arguments: 2
Explanation: The expression that is the value of the second argument is evaluated in an empty environment. The number of elements of the first argument gives a time limit (that is, a maximum number of computations equal to the length of the list or zero if the first argument is not a list). The time limit actually limits the depth of the evaluation. If the evaluation completes within the time limit, the value returned is a list whose sole element is the value of the value of the second argument. If the evaluation is not completed within the time limit, the value returned is the atom for “Try.”

14 Symbol: [&]
Name: Define Function or Lambda
Arguments: 2
Explanation: Treated essentially as a primitive function, this atom is used to create a defined function where the first argument is a list of variables and the second argument is the body of the function definition. Note that all other (non-reserved) symbols may be used as variables in a defined function.

We extend LISP to define a self-delimiting universal computer. The computer's program appears on its tape as a binary representation of a LISP expression. Note that the program must be self-delimiting because the S-expression must have balanced delimiters.

13 We redefine "Try" by adding an argument to be able to initially place information on the computer's tape. The three arguments are as follows. The first argument, the depth-limit, is altered from the original LISP definition: if it is a non-null atom, then there is no depth limit; if it is the empty list, there is zero depth limit (that is, no function calls or re-evaluations); if it is an n-element list, there is a depth limit of n. The second argument is as before: the expression to be evaluated as long as the depth limit is not exceeded. The new third argument is a list of bits to be used as the computer's program tape.

13 The value returned by "Try" is also changed. If the computation terminates normally, the first element of the returned value is a list with only one element, which is the result of the computation. If the evaluation of the second argument aborts, the first element of the
12 returned value is the atom for "Evaluate" after an attempt to read a non-existent bit from the
13 tape or the atom for "Try" when the number of computations exceeds the depth limit. The rest
10 of the returned value is a stack of all the arguments to the primitive function "Display" encountered during the evaluation of the second argument.

We reserve two more symbols for primitives that could be programmed but are built-in to help conveniently define and efficiently run a self-delimiting universal computer.

15 Symbol: [^]
Name: Append
Arguments: 2
Explanation: The result of this function is the concatenation of its two arguments into a single list.

16 Symbol: [%]
Name: Read-Expression
Explanation: Read an entire LISP expression from the computer's tape. This function is the only way that information on the computer's tape can be accessed. It must be implemented in a self-delimiting fashion because no algorithm can search for the end of the tape and then use the length of the tape as data in the computation. If an algorithm attempts to read a bit that is not on the tape, the algorithm aborts. That is, this function explodes if the tape is exhausted, killing the computation.

In conclusion, permissiveness in our LISP is achieved because functions with extra arguments are evaluated but ignored and empty lists are supplied for missing arguments. There are no erroneous expressions; only expressions that never return a value because the interpreter goes into an infinite loop.

Approximating Omega:

Section 2:

```
((''(&(V)((''(&(A)((''(&(R)((''(&(W)(W(''(O)))))(''(&(n)(*0(*.(R(Vn())n)))))))))(''(&(xy)(/(.x)(/(.y)()(*0(Rx(-
y))))(^(R(-x)(y))(*(+x)())))))))))(''(&(xyz)(/(.x)(/(.y)(/z(''(1)))()(A(''(0))yz))(/(.y)(Ax(''(0))z)(*(=(+x)
(=(+y)z))(A(-x)(-y)(/(+x)(/(+y)1z)(/(+y)z0)))))))))))(''(&(xy)(/(.x)(/(.(+(?n(''(!(%)))y))))()(''(1)))(A(V(
-x)(*0y))(V(-x)(*1y))0)))))
```

Above is the program approximating Omega given in its ascii representation. Note that the symbol O represents a list of 1s with a length that determines how many bits of the binary expansion of Omega are approximated.

Part 0 (time-unit, sound):

0,	0	26,	4	52,	3	78,	22	104,	1	130,	1	156,	23	182,	0	208,	6	234,	0	260,	0	286,	12	312,	1
1,	0	27,	0	53,	0	79,	23	105,	1	131,	1	157,	1	183,	11	209,	0	235,	7	261,	4	287,	0	313,	0
2,	4	28,	14	54,	9	80,	1	106,	1	132,	1	158,	0	184,	0	210,	7	236,	23	262,	0	288,	16	314,	9
3,	0	29,	0	55,	5	81,	0	107,	1	133,	1	159,	11	185,	5	211,	23	237,	1	263,	14	289,	1	315,	3
4,	14	30,	20	56,	0	82,	11	108,	0	134,	1	160,	24	186,	23	212,	1	238,	2	264,	0	290,	1	316,	23
5,	0	31,	1	57,	19	83,	0	109,	15	135,	1	161,	0	187,	1	213,	24	239,	24	265,	22	291,	1	317,	1
6,	17	32,	0	58,	0	84,	5	110,	0	136,	1	162,	4	188,	0	214,	1	240,	1	266,	23	292,	23	318,	1
7,	1	33,	20	59,	17	85,	22	111,	19	137,	0	163,	0	189,	18	215,	1	241,	0	267,	1	293,	1	319,	0
8,	0	34,	0	60,	25	86,	1	112,	0	138,	4	164,	2	190,	22	216,	0	242,	11	268,	0	294,	1	320,	17
9,	0	35,	4	61,	0	87,	0	113,	8	139,	0	165,	1	191,	0	217,	18	243,	0	269,	11	295,	1	321,	0
10,	4	36,	0	62,	1	88,	11	114,	22	140,	14	166,	1	192,	4	218,	0	244,	7	270,	0	296,	0	322,	8
11,	0	37,	21	63,	1	89,	0	115,	1	141,	0	167,	0	193,	0	219,	8	245,	23	271,	5	297,	1	323,	22
12,	14	38,	1	64,	25	90,	5	116,	0	142,	22	168,	1	194,	3	220,	22	246,	1	272,	22	298,	0	324,	1
13,	0	39,	1	65,	1	91,	23	117,	8	143,	23	169,	1	195,	1	221,	1	247,	24	273,	1	299,	4	325,	0
14,	18	40,	1	66,	1	92,	1	118,	23	144,	24	170,	0	196,	1	222,	0	248,	3	274,	0	300,	0	326,	9
15,	1	41,	1	67,	1	93,	0	119,	1	145,	1	171,	18	197,	24	223,	8	249,	1	275,	11	301,	2	327,	2
16,	0	42,	1	68,	1	94,	1	120,	1	146,	0	172,	0	198,	1	224,	23	250,	1	276,	0	302,	1	328,	23
17,	0	43,	0	69,	1	95,	0	121,	0	147,	11	173,	4	199,	0	225,	1	251,	1	277,	5	303,	1	329,	1
18,	4	44,	4	70,	1	96,	9	122,	9	148,	0	174,	0	200,	9	226,	0	252,	1	278,	0	304,	1	330,	1
19,	0	45,	0	71,	1	97,	3	123,	0	149,	5	175,	3	201,	0	227,	11	253,	1	279,	7	305,	0	331,	3
20,	14	46,	14	72,	1	98,	0	124,	7	150,	22	176,	1	202,	6	228,	0	254,	1	280,	0	306,	18	332,	1
21,	0	47,	0	73,	0	99,	19	125,	22	151,	1	177,	1	203,	0	229,	7	255,	1	281,	13	307,	0	333,	1
22,	19	48,	25	74,	4	100,	22	126,	1	152,	0	178,	23	204,	7	230,	22	256,	1	282,	25	308,	17	334,	1
23,	1	49,	1	75,	0	101,	0	127,	0	153,	11	179,	24	205,	22	231,	1	257,	1	283,	0	309,	0	335,	1
24,	0	50,	0	76,	14	102,	8	128,	1	154,	0	180,	1	206,	1	232,	0	258,	1	284,	4	310,	8	336,	1
25,	0	51,	9	77,	0	103,	23	129,	1	155,	5	181,	1	207,	0	233,	11	259,	1	285,	0	311,	22		

Assign each part played by a pitched instrument a distinct pitch class from the following:
(D+0, A+2, F#-14, C-31, G#-49, A#+41, D#+5, F-2, G#+28, C+30, C#+45, F-49, F#+29, G+12, A-34, B-26, C#-41)

Option 1 (start time-unit, end-time unit):

Part 1	Part 2	Part 3	Part 4	Part 5	Part 6	Part 7	Part 8	Part 9	Part 10	Part 11	Part 12	Part 13	Part 14	Part 15	Part 16	Part 17
0,336	1,259	3,258	5,7	9,136	11,135	13,15	17,72	19,71	21,23	25,42	27,41	29,31	34,39	36,38	58,63	61,62
	260,335	262,334	8,257	137,256	139,255	16,134	73,133	75,132	24,70	43,69	45,68	32,40	53,66	56,65		
			264,267	270,273	276,295	141,145	148,151	154,157	77,80	83,86	89,92	47,49	101,104			
			268,333	274,304	296,297	146,254	152,181	158,169	81,131	87,107	93,94	50,67				
				305,332	298,303	278,294	182,253	170,180	161,166	108,130	95,106	98,105				
					307,318	300,302	280,293	184,187	167,168	163,165	110,120	112,115				
					319,330	309,312		188,198	172,177	174,176	121,129	116,119				
						313,317		199,252	191,196	193,195	209,212	123,126				
						321,324		283,291	201,215	203,206	228,231	127,128				
						325,329			216,251	207,214	232,240	234,237				
									285,290	218,221	241,249	243,246				
										222,225						
										226,250						
										287,289						

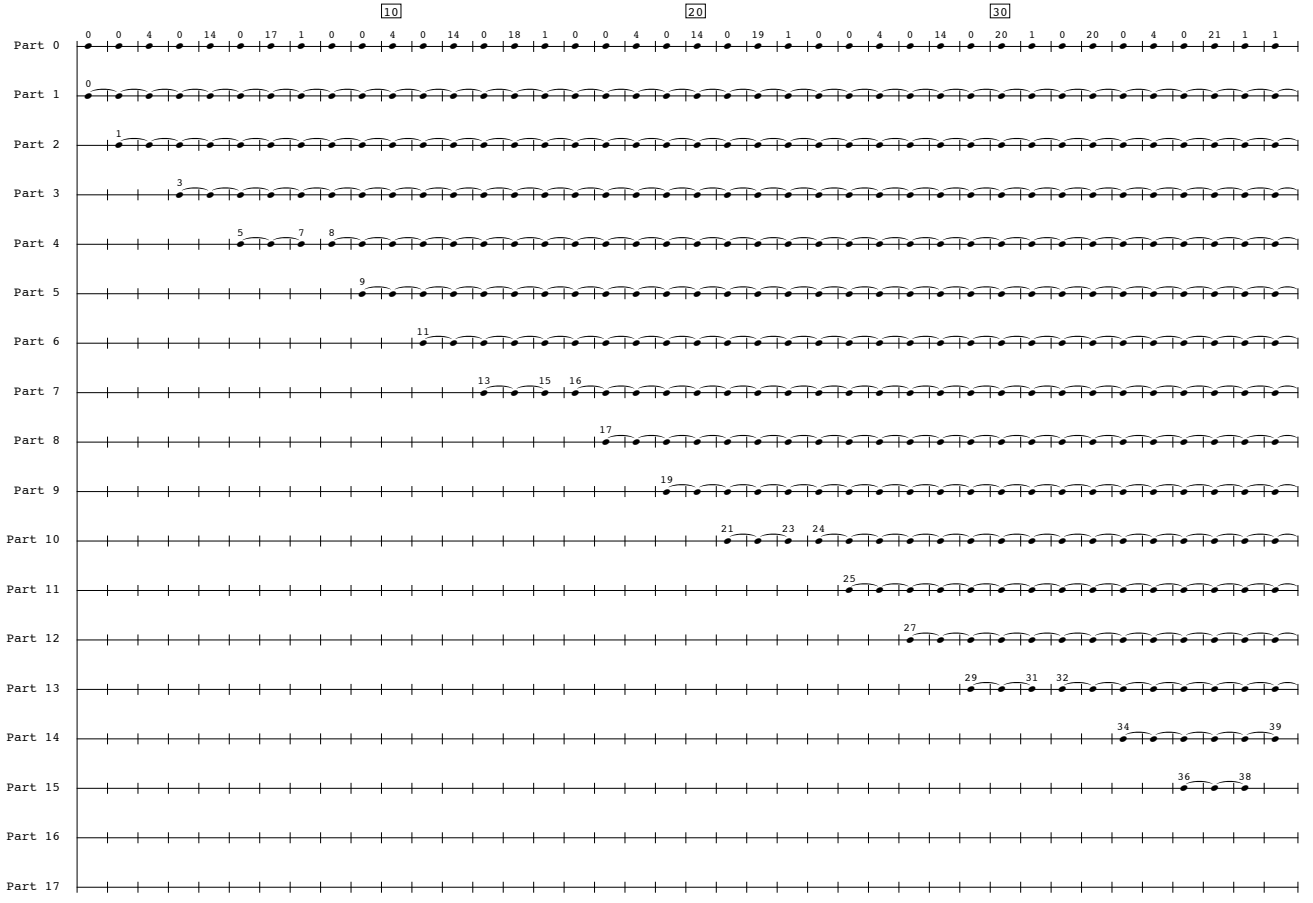
Option 2 (start time-unit, end-time unit):

Part 1	Part 2	Part 3	Part 4	Part 5	Part 6	Part 7	Part 8	Part 9	Part 10	Part 11	Part 12	Part 13	Part 14	Part 15	Part 16	Part 17
0,336	1,259	3,258	5,7	8,257	9,136	11,135	13,15	167,168	17,72	19,71	21,23	24,70	25,42	27,41	29,31	32,40
	305,332	300,302	34,39	298,303	172,177	170,180	36,38	193,195	98,105	95,106	43,69	93,94	58,63	56,65	45,68	53,66
			50,67		203,206	207,214	47,49	218,221	121,129	123,126	89,92	112,115	75,132	77,80	87,107	81,131
			83,86		228,231	234,237	61,62	243,246	152,181	146,254	110,120	137,256	161,166	101,104	127,128	158,169
			108,130		262,334	264,267	73,133	270,273	209,212	285,290	141,145	296,297	191,196	116,119	148,151	199,252
			154,157			307,318	163,165	309,312	232,240	319,330	174,176		222,225	139,255	182,253	280,293
			184,187				188,198		268,333		201,215		260,335	287,289	283,291	325,329
			216,251				226,250				241,249			313,317	321,324	
			278,294				276,295				274,304					

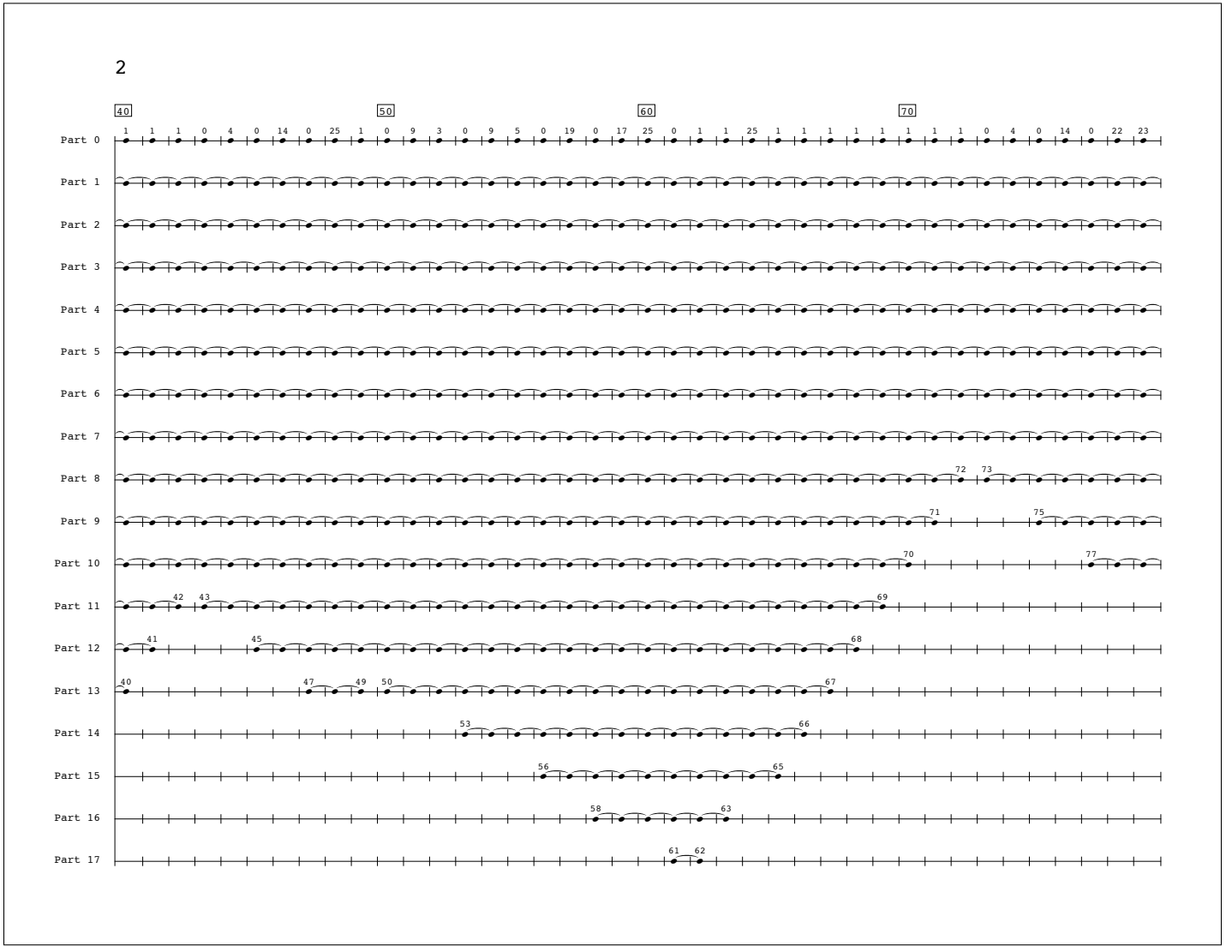
Appendix
Sustained Tones (start time-unit, end-time unit):
((0,336),(1,259),(3,258),(5,7),(8,257),(9,136),(11,135),(13,15),(16,134),(17,72),(19,71),(21,23),
(24,70),(25,42),(27,41),(29,31),(32,40),(34,39),(36,38),(43,69),(45,68),(47,49),(50,67),(53,66),
(56,65),(58,63),(61,62),(73,133),(75,132),(77,80),(81,131),(83,86),(87,107),(89,92),(93,94),(95,106),
(98,105),(101,104),(108,130),(110,120),(112,115),(116,119),(121,129),(123,126),(127,128),(137,256),
(139,255),(141,145),(146,254),(148,151),(152,181),(154,157),(158,169),(161,166),(163,165),(167,168),
(170,180),(172,177),(174,176),(182,253),(184,187),(188,198),(191,196),(193,195),(199,252),(201,215),
(203,206),(207,214),(209,212),(216,251),(218,221),(222,225),(226,250),(228,231),(232,240),(234,237),
(241,249),(243,246),(260,335),(262,334),(264,267),(268,333),(270,273),(274,304),(276,295),(278,294),
(280,293),(283,291),(285,290),(287,289),(296,297),(298,303),(300,302),(305,332),(307,318),(309,312),
(313,317),(319,330),(321,324),(325,329));

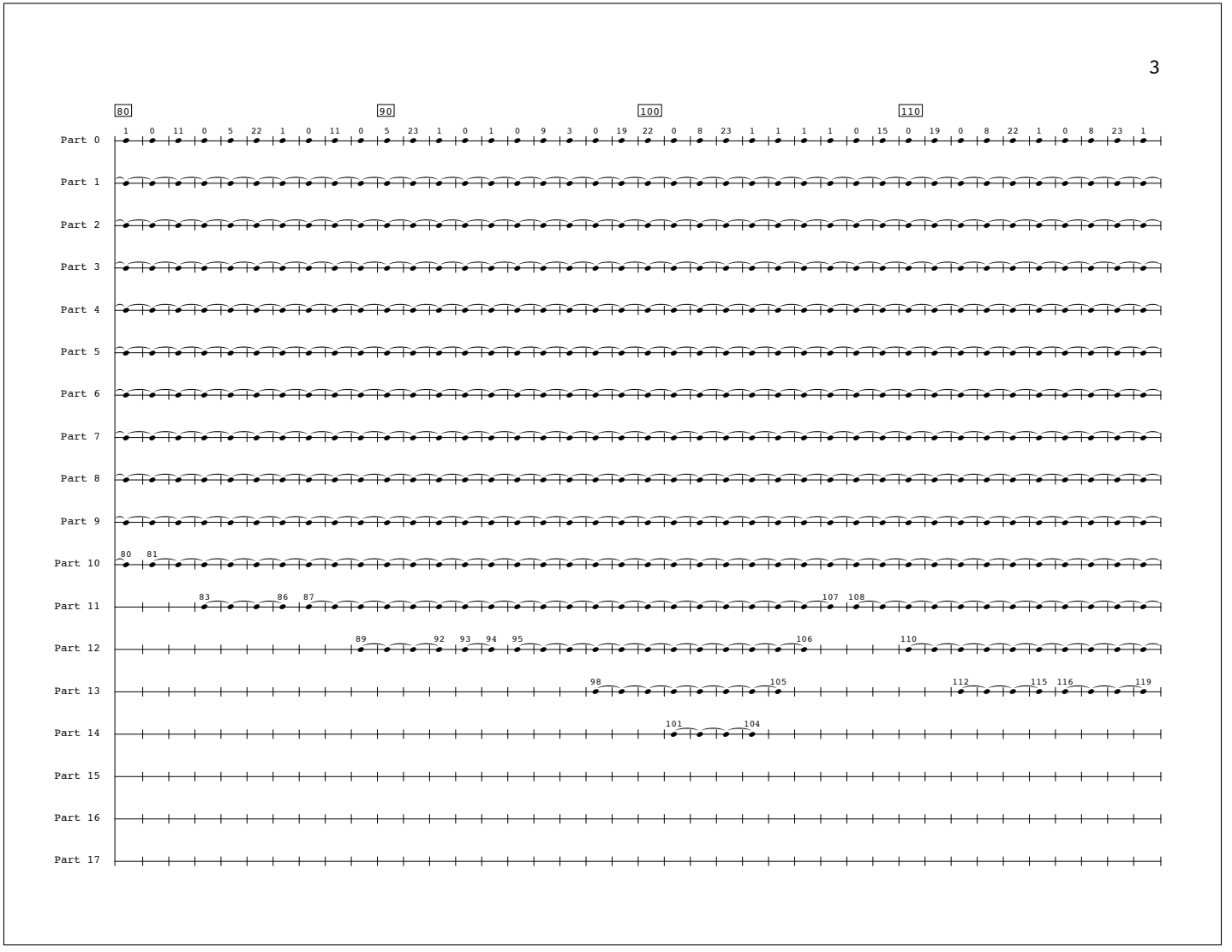
Approximating Omega
Section 2; Option 1

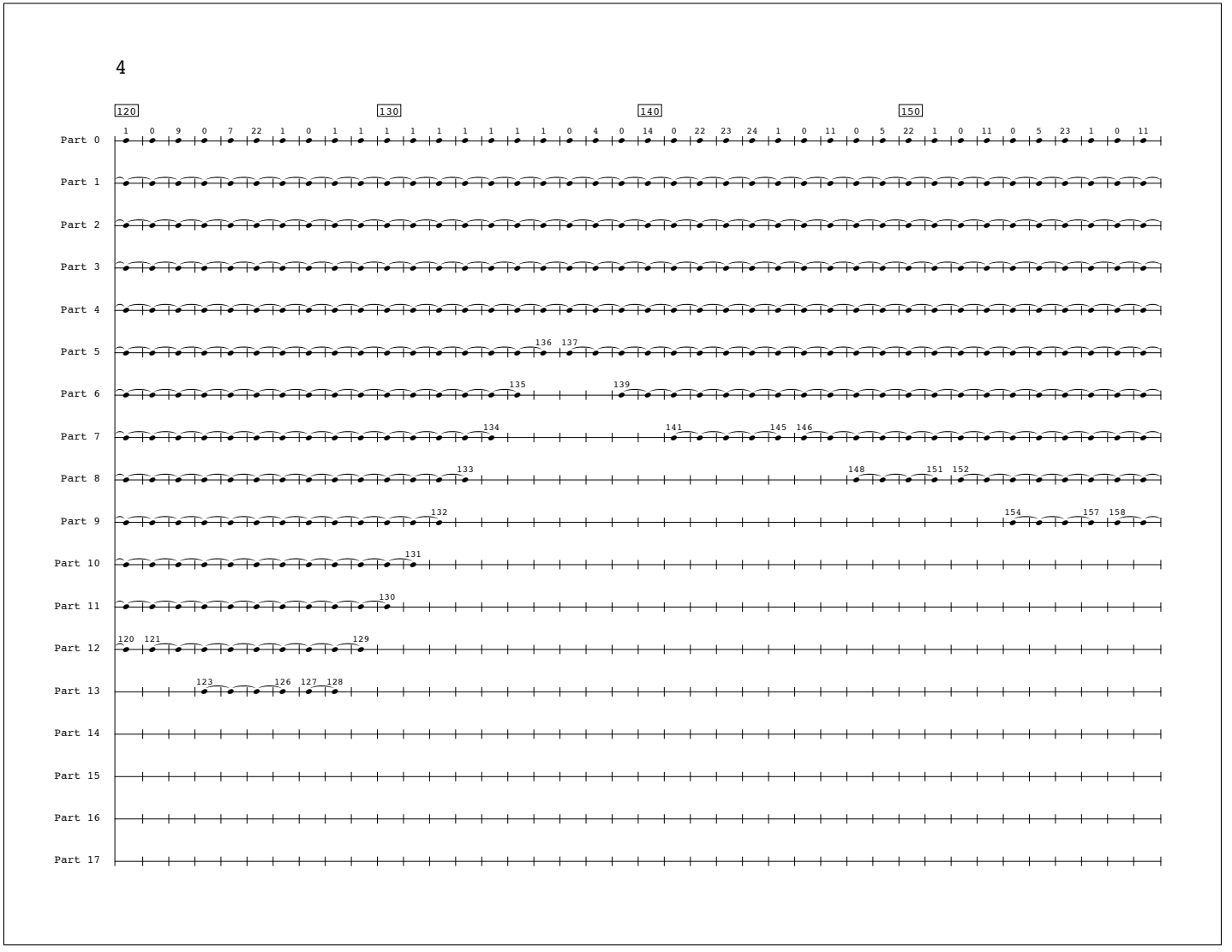
Assign each part from 1 through 17 played by a pitched instrument a distinct pitch class from the following (each tone may be played in any octave):
(D+0, A+2, F#-14, C-31, G#-49, A#+41, D#+5, F-2, G#+28, C+30, C#+45, F-49, F#+29, G+12, A-34, B-26, C#-41)

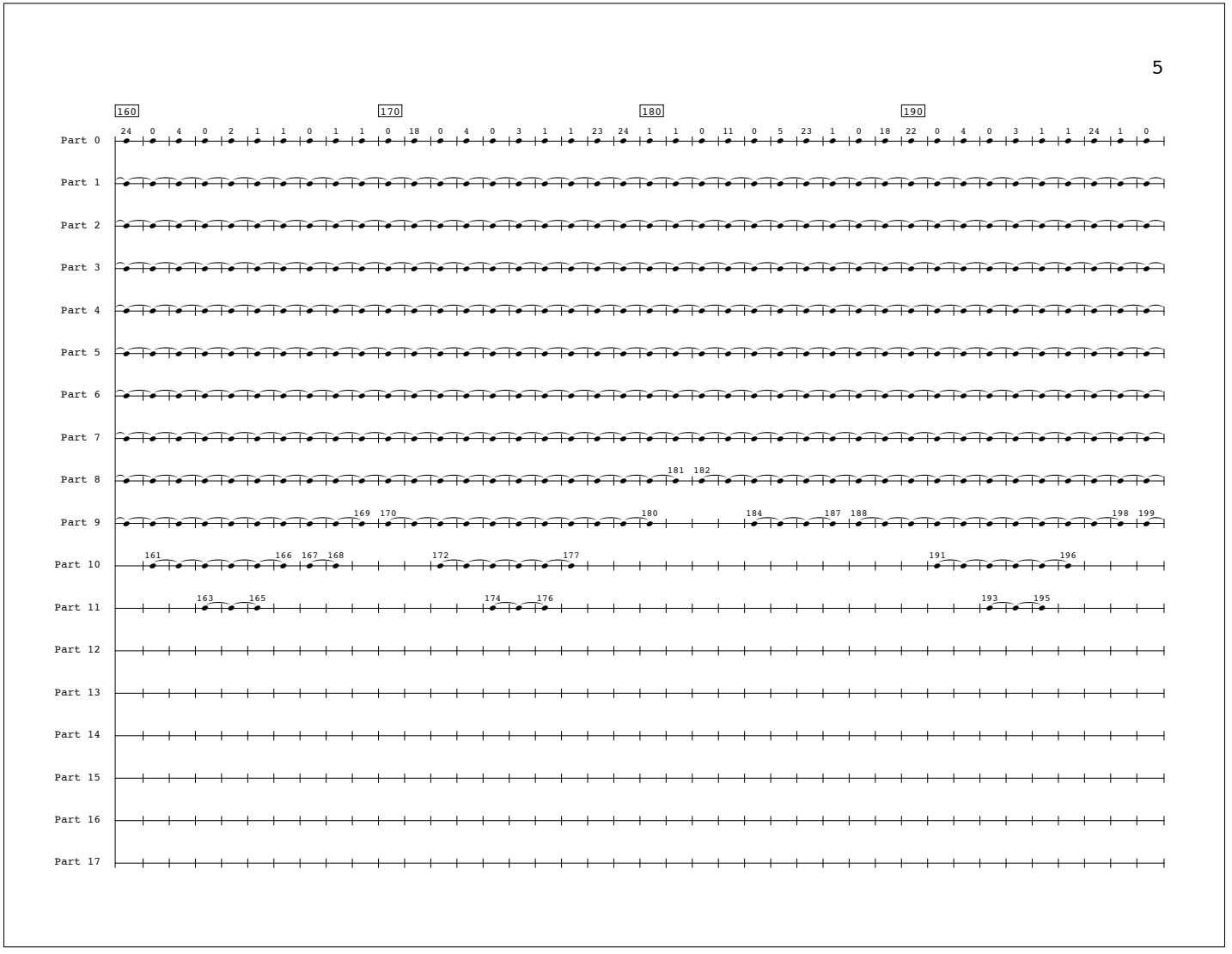


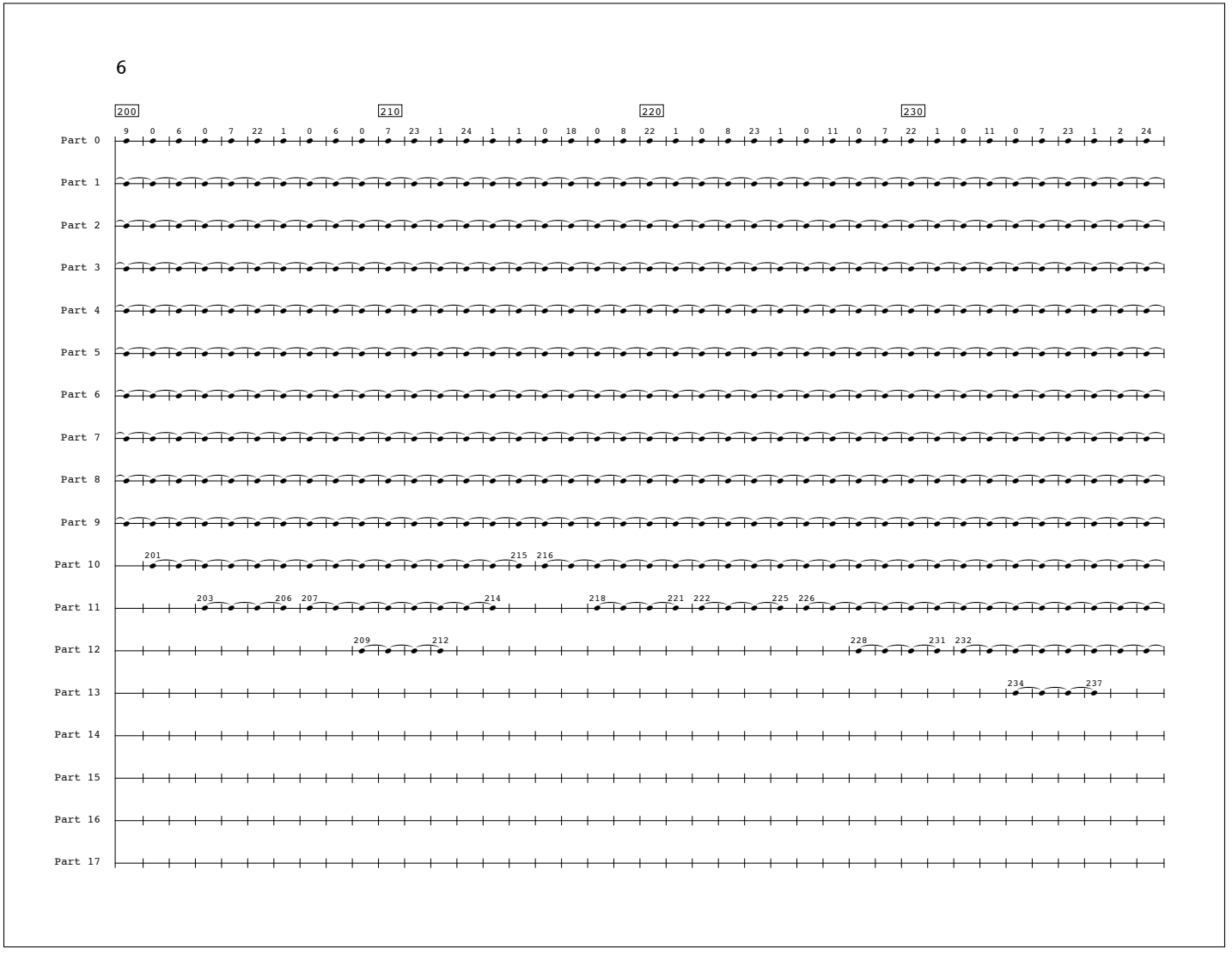
*Part 0: The number above each note indicates which sound to play. The sound may occur at any point in the time unit (or measure).
*Parts 1 through 17: The numbers above the starts and ends of each note indicate the time-unit (or measure) of entry or exit. These should coincide presicely with the sound occuring in part 1, which may occur at any point in the time-unit. Note that measure numbers start from index 0.

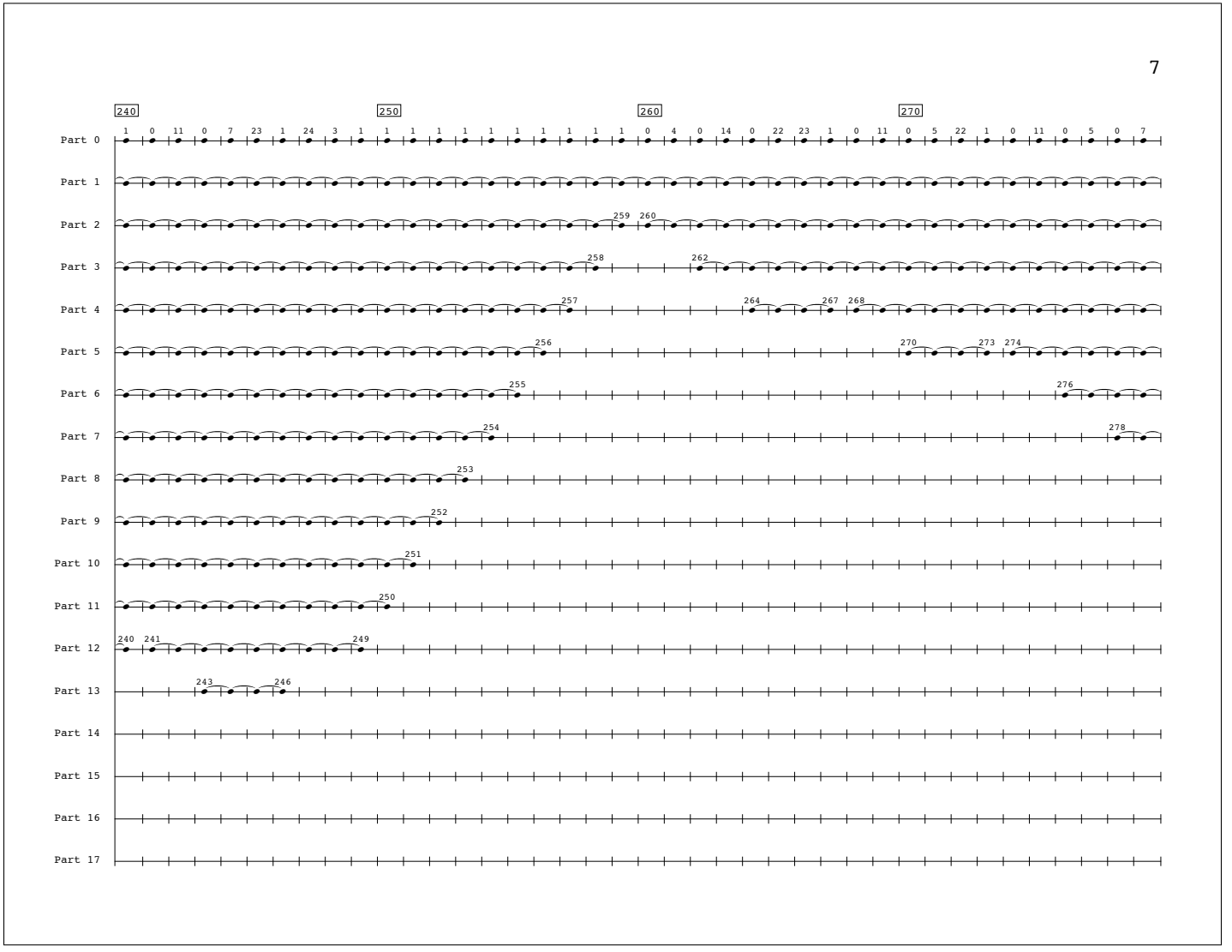

2
40
50
60
70
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
Part 9
Part 10
Part 11
Part 12
Part 13
Part 14
Part 15
Part 16
Part 17

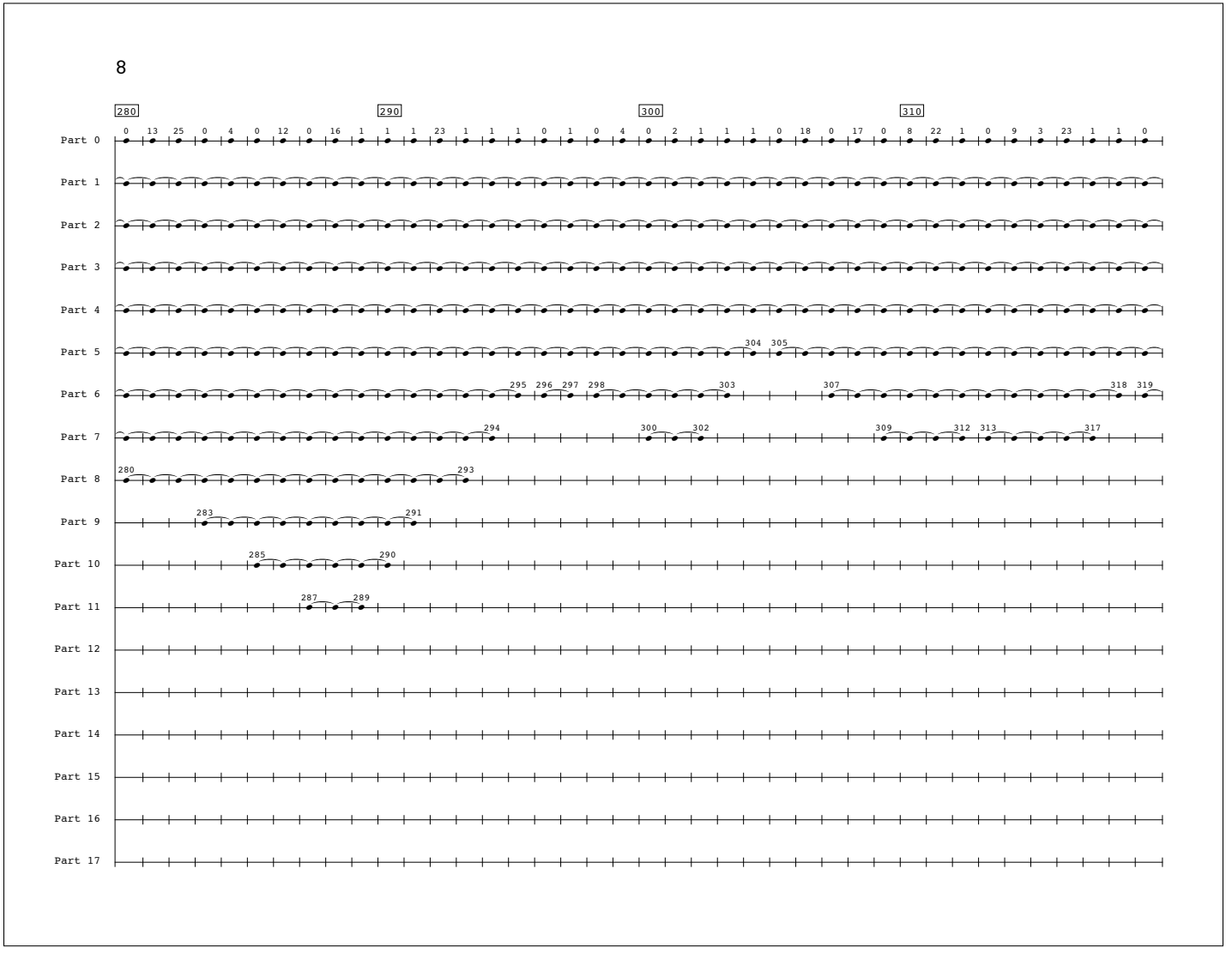

80
90
100
110
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
Part 9
Part 10
Part 11
Part 12
Part 13
Part 14
Part 15
Part 16
Part 17

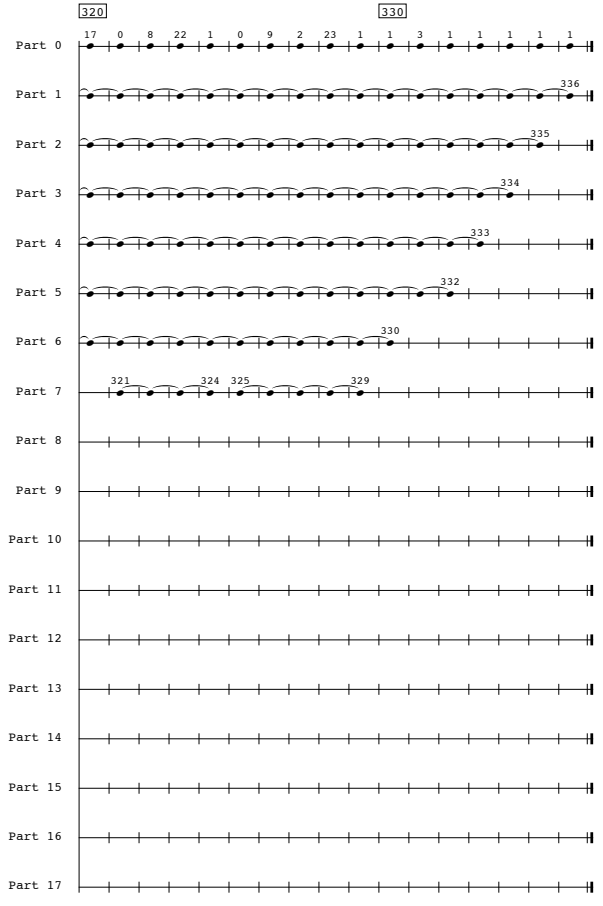

4
120
130
140
150
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
Part 9
Part 10
Part 11
Part 12
Part 13
Part 14
Part 15
Part 16
Part 17


5
160
170
180
190
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
Part 9
Part 10
Part 11
Part 12
Part 13
Part 14
Part 15
Part 16
Part 17


6
200
210
220
230
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
Part 9
Part 10
Part 11
Part 12
Part 13
Part 14
Part 15
Part 16
Part 17


240
250
260
270
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
Part 9
Part 10
Part 11
Part 12
Part 13
Part 14
Part 15
Part 16
Part 17

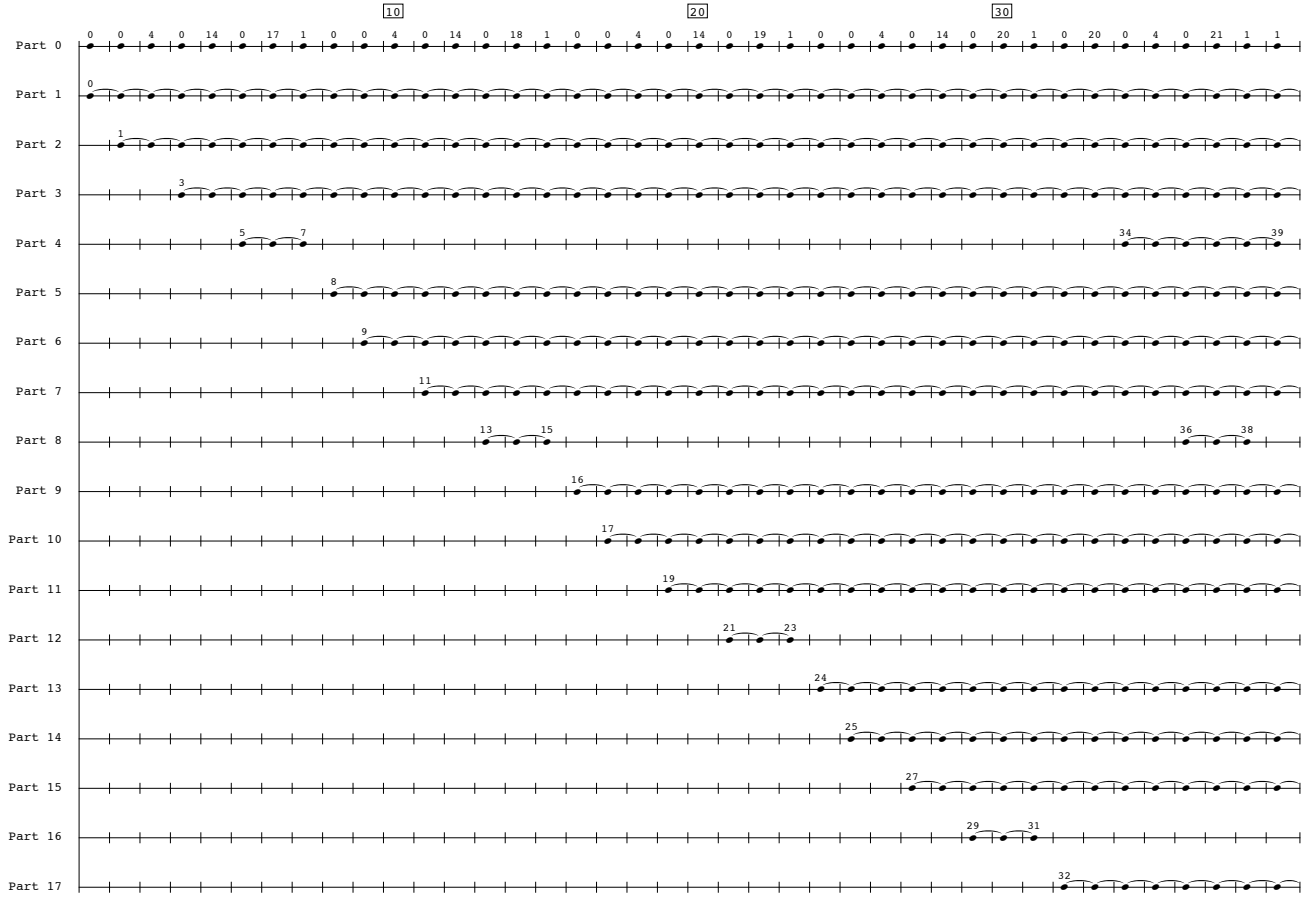

8
280
290
300
310
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
Part 9
Part 10
Part 11
Part 12
Part 13
Part 14
Part 15
Part 16
Part 17



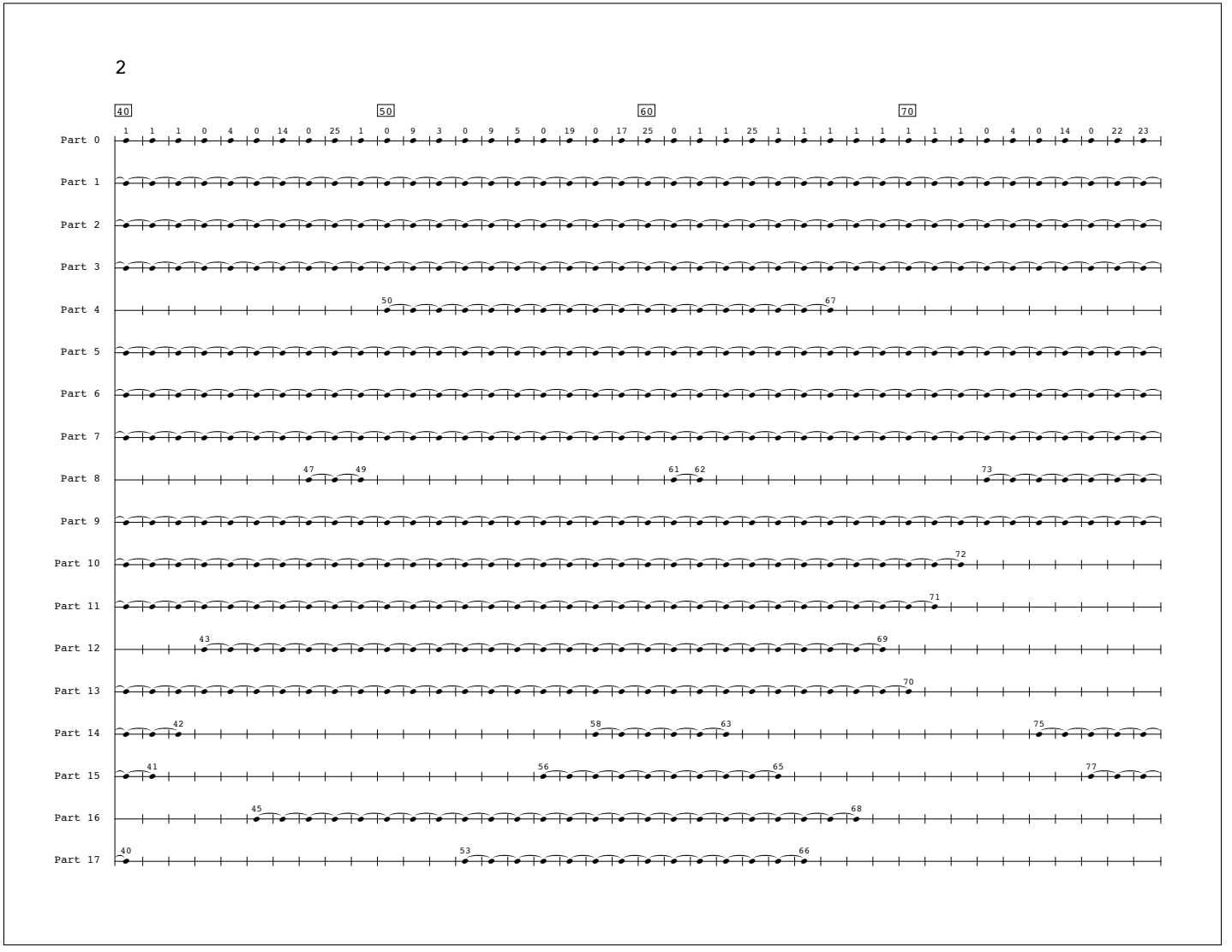
-michael winter (la; 2010)

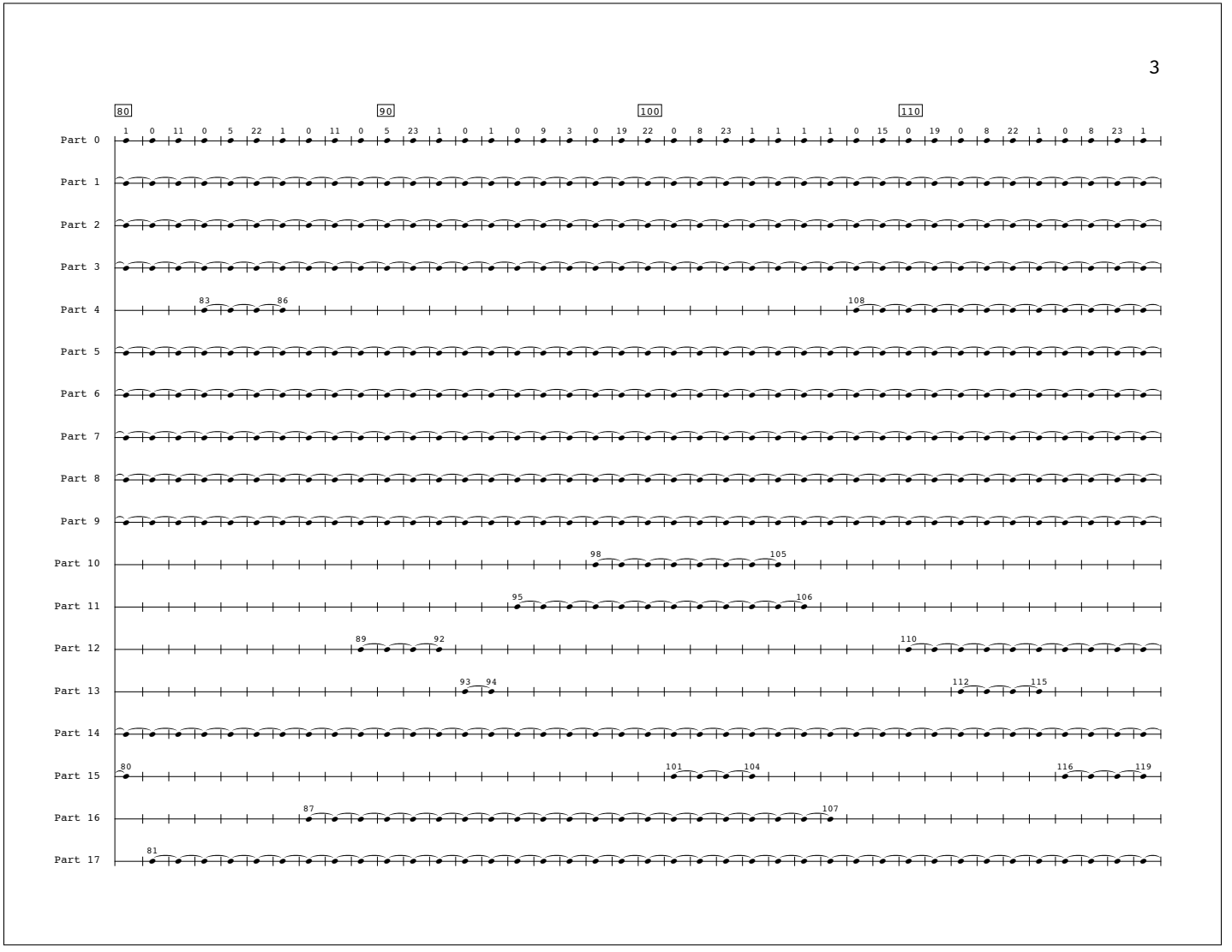
Approximating Omega
Section 2; Option 2

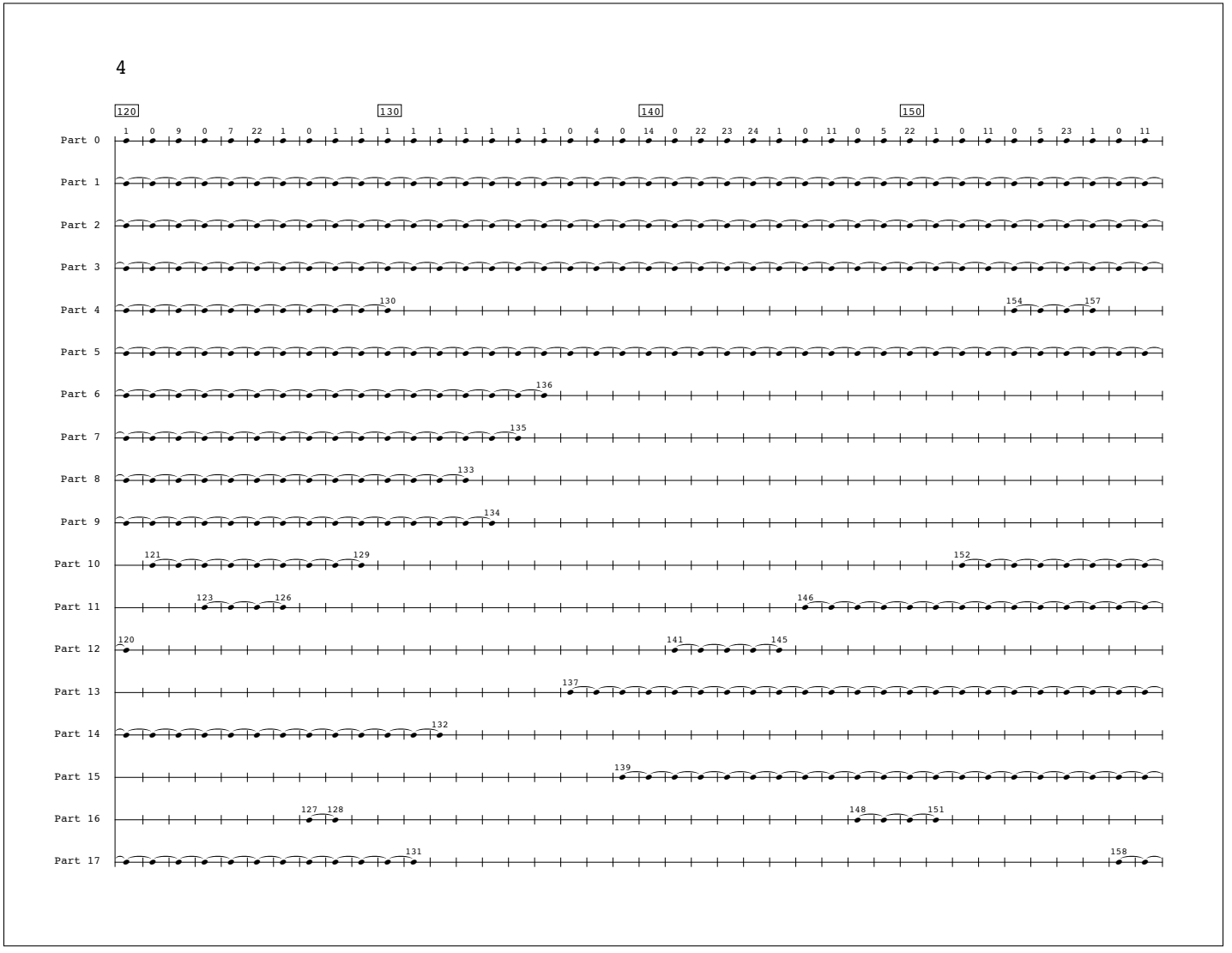
*Assign each part from 1 through 17 played by a pitched instrument a distinct pitch-class from the following (each tone may be played in any octave):
(D+0, A+2, F#-14, C-31, G#-49, A#+41, D#+5, F-2, G#+28, C+30, C#+45, F-49, F#+29, G+12, A-34, B-26, C#-41)

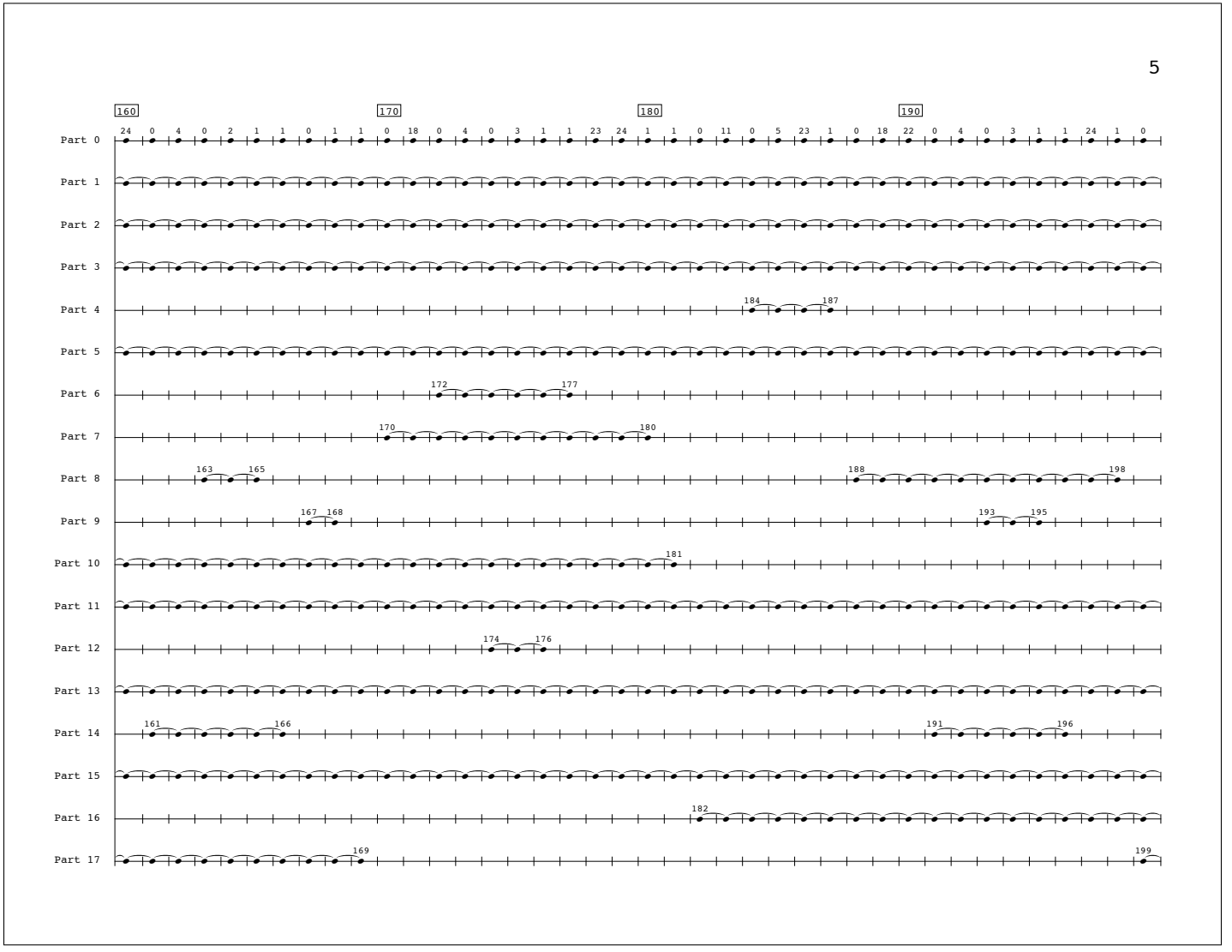


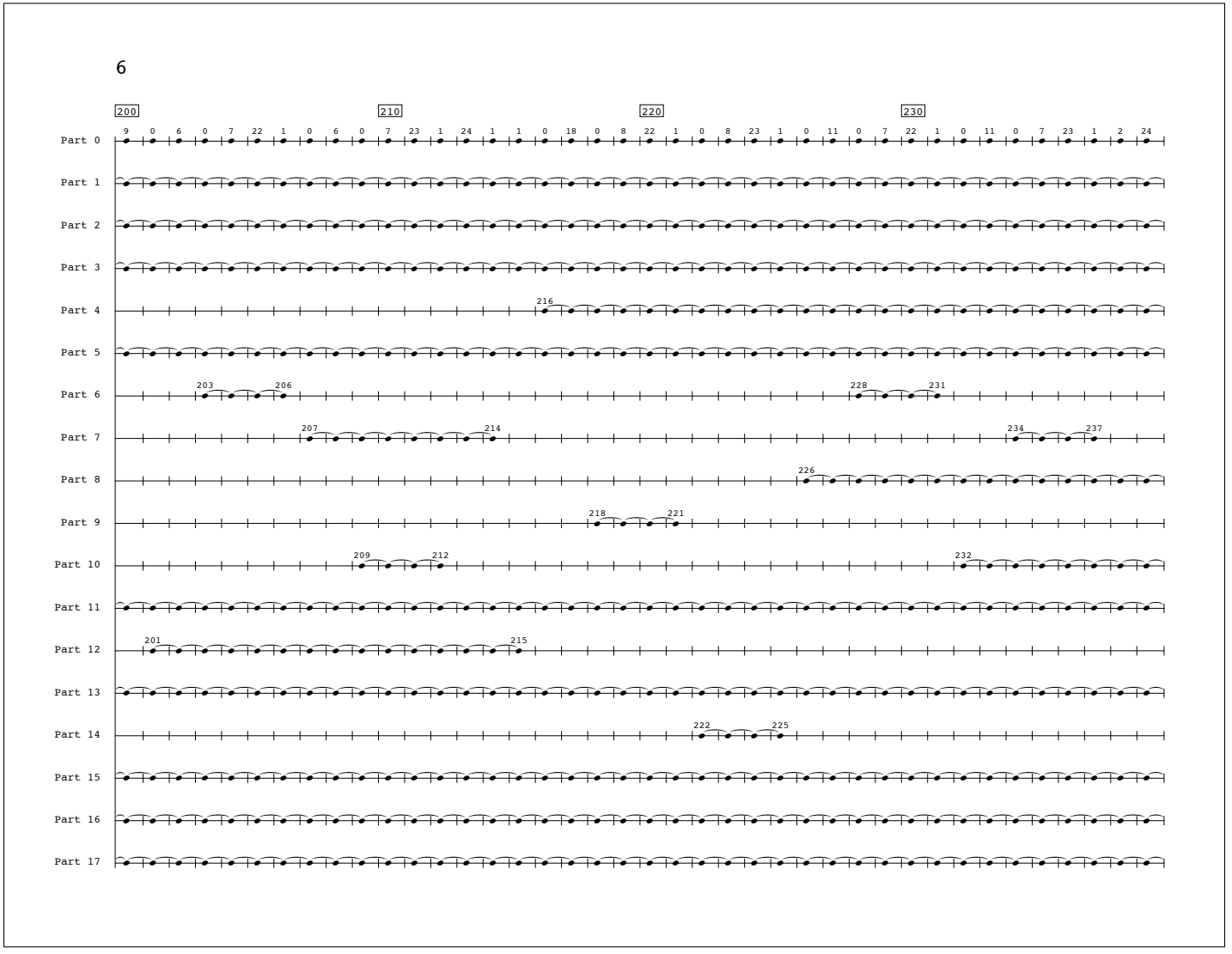
*Part 0: The number above each note indicates which sound to play. The sound may occur at any point in the time unit (or measure).
*Parts 1 through 17: The numbers above the starts and ends of each note indicate the time-unit (or measure) of entry or exit. These should coincide presicely with the sound occuring in part 1, which may occur at any point in the time-unit. Note that measure numbers start from index 0.

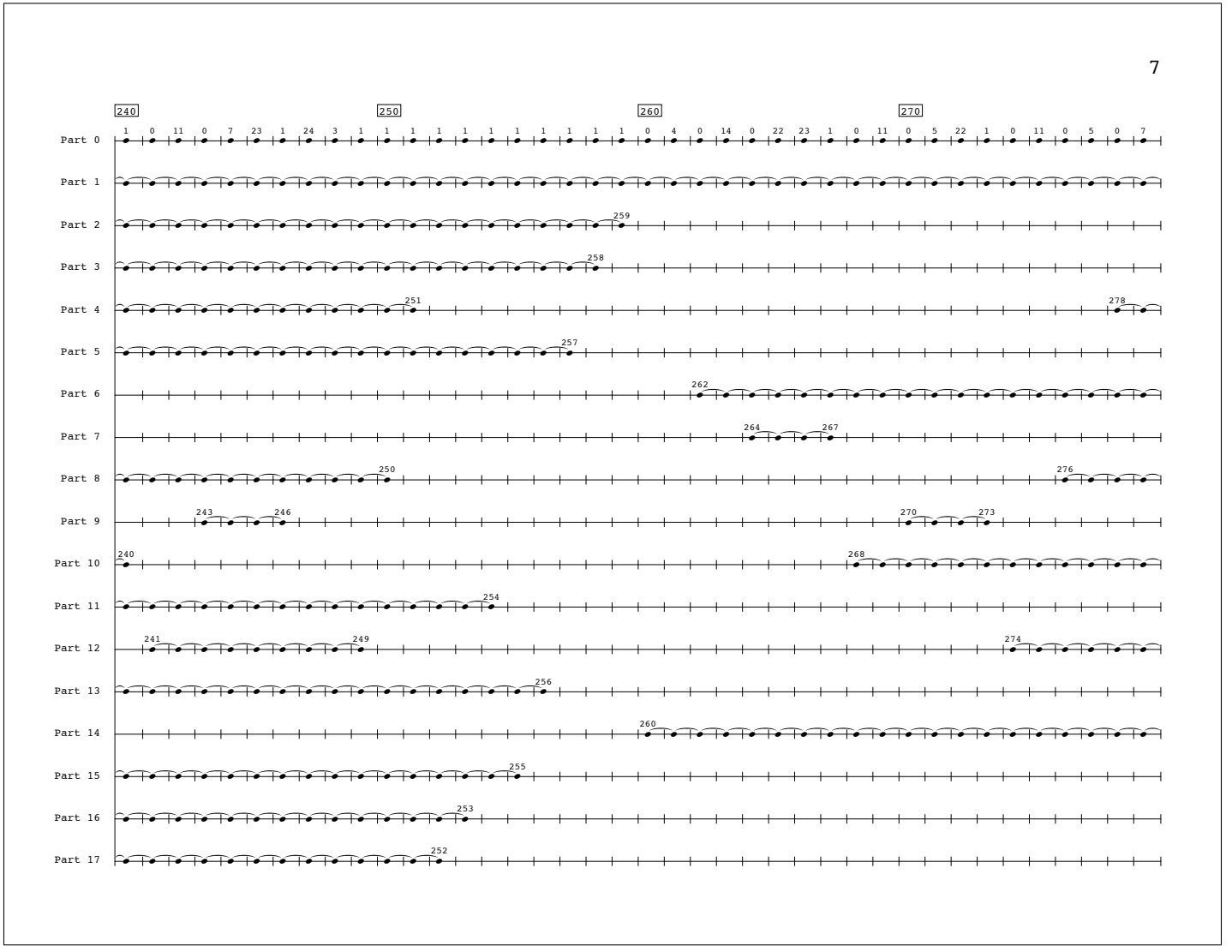

2
40
50
60
70
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
Part 9
Part 10
Part 11
Part 12
Part 13
Part 14
Part 15
Part 16
Part 17

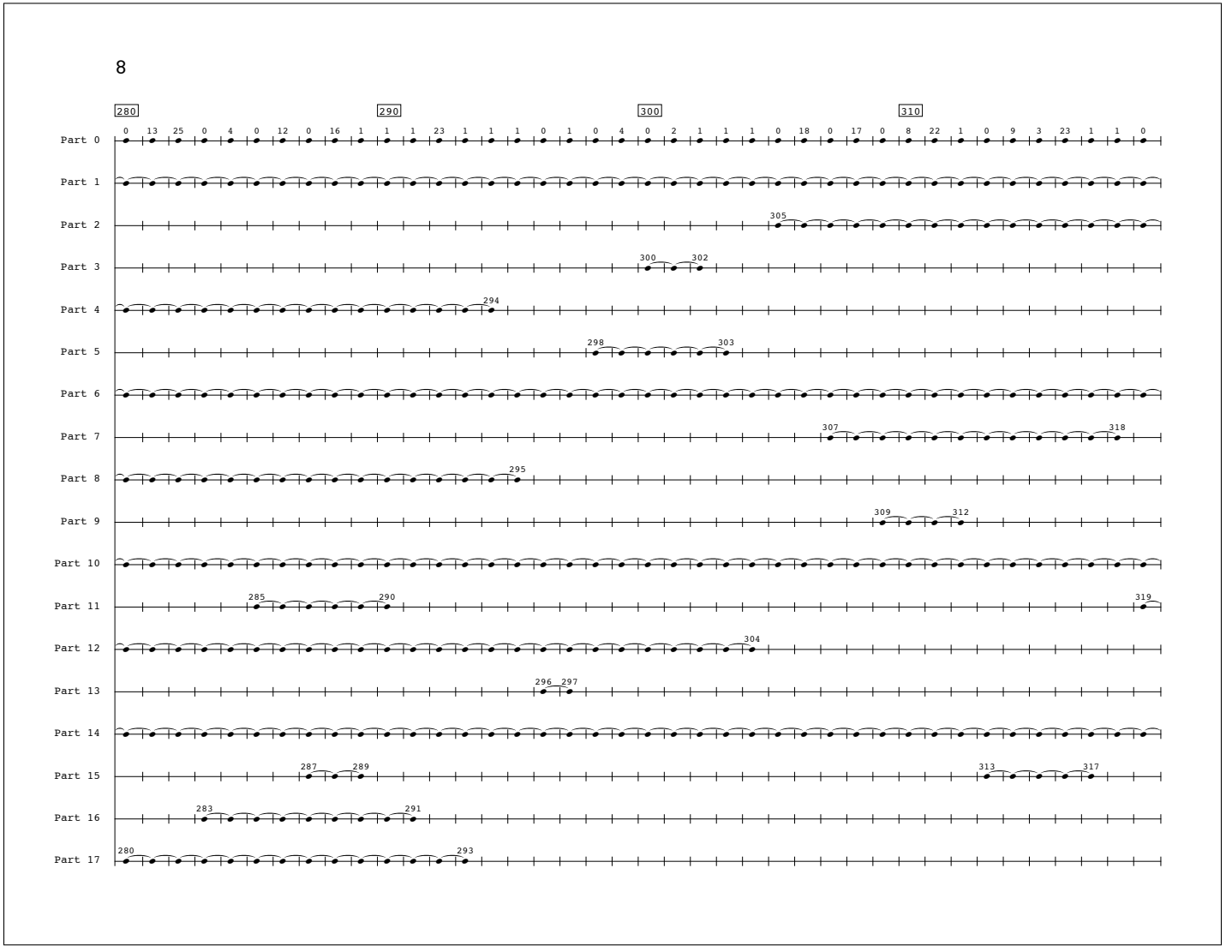

80
90
100
110
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
Part 9
Part 10
Part 11
Part 12
Part 13
Part 14
Part 15
Part 16
Part 17

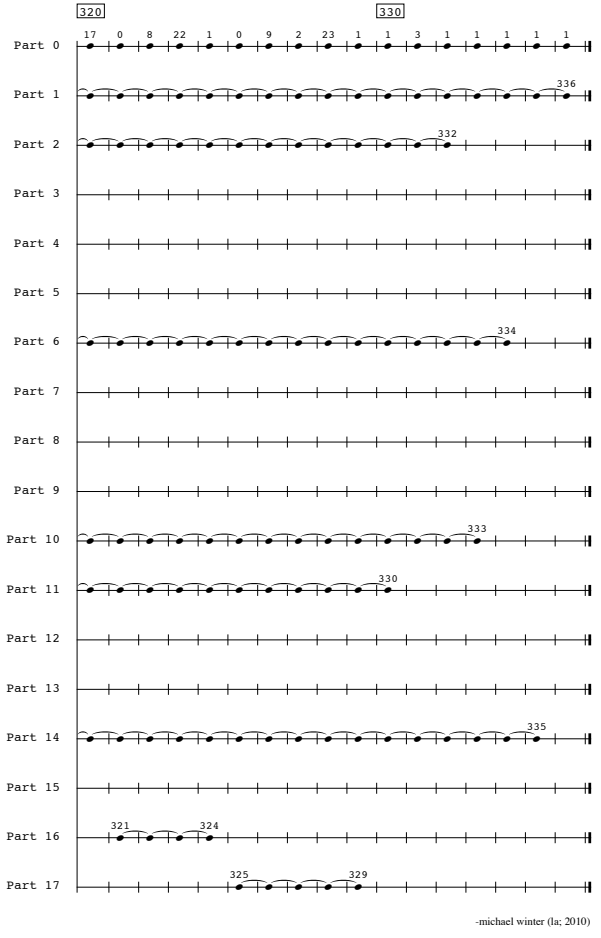

4
120
130
140
150
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
Part 9
Part 10
Part 11
Part 12
Part 13
Part 14
Part 15
Part 16
Part 17


160
170
180
190
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
Part 9
Part 10
Part 11
Part 12
Part 13
Part 14
Part 15
Part 16
Part 17


6
200
210
220
230
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
Part 9
Part 10
Part 11
Part 12
Part 13
Part 14
Part 15
Part 16
Part 17


240
250
260
270
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
Part 9
Part 10
Part 11
Part 12
Part 13
Part 14
Part 15
Part 16
Part 17


8
280
290
300
310
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
Part 9
Part 10
Part 11
Part 12
Part 13
Part 14
Part 15
Part 16
Part 17



-michael winter (la; 2010)

A.11 *pedal, triangle machine, and (perhaps) coda*

pedal, triangle machine, and (perhaps) coda consists of a long drone interspersed with with short flurries of percussion activity with an optional computer synthesized coda at the end. The triangle parts are generated by following the state sequences of Turing machines chosen at random. The current state of two Turing machines determine the duration between each attack and whether or not the triangle is played opened or stopped.

While several triangle parts were generated, only one is presented here. Each part was chosen based on whether I perceived them as chaotic for the duration of the part. If the Turing machine quickly started looping or repeating, I would discard that part choosing only runs that seemed random. Here the metric is essentially my perception of randomness. Since I was only interested in the first several hundred states of the output, any of these machines may very well get in a loop after several more states beyond where I was looking.

pedal, triangle machine, and (perhaps) coda

for casey thomas anderson

performers distributed throughout the space to the extent possible. 17 minutes or more in total. if performed in a very large space, or in several rooms, run multiple, independent realizations simultaneously throughout the space.

pedal:

a drone with a loud constancy throughout the piece. preferably low horns (primarily trombones and tubas, if possible) with staggered breaths. pitch-class 'a' with each tone slightly detuned. (the start of the drone may be accented with a percussive attack that decays slowly. the end may be accented with a percussive attack that is punctuated.)

auxiliary pedal (optional):

for 5 or more pitched or unpitched sustaining sounds that blend with, and subtly color, the drone. divide the entire length of the piece into equal sections greater than 3 minutes. a different group of 3 or (preferably) more performers play throughout each section as follows: sound a tone followed by silence; then repeat the sound and silence, shortening or lengthening the duration of one or the other by a small amount. the duration of each sound/silence should be between 3 and 11 seconds (perhaps quantized to a strict time-subdivision scheme). each pitched sound must be an octave equivalent of a unique prime harmonic (no doubling pitch-classes) of an 'a,' with each tone slightly detuned. each unpitched sound must have a distinct timbre from all others. (the start and end of each section may be accented with a percussive attack that is punctuated.)

triangle machine:

each occurrence consists of 3, 5, or 7 performers each playing a triangle (or triangle-like instrument, preferably of the same type/sound for all) and reading from a unique part (provided on the following pages). the players start together and maintain a strict tempo throughout such that a measure lasts 1 to 2 seconds (thus each occurrence lasts 2 to 4 minutes in total). a cross indicates that the triangle is played stopped. a circle above a note indicates that the triangle is played open and allowed to decay naturally till the next attack or silence (despite any rests that follow).

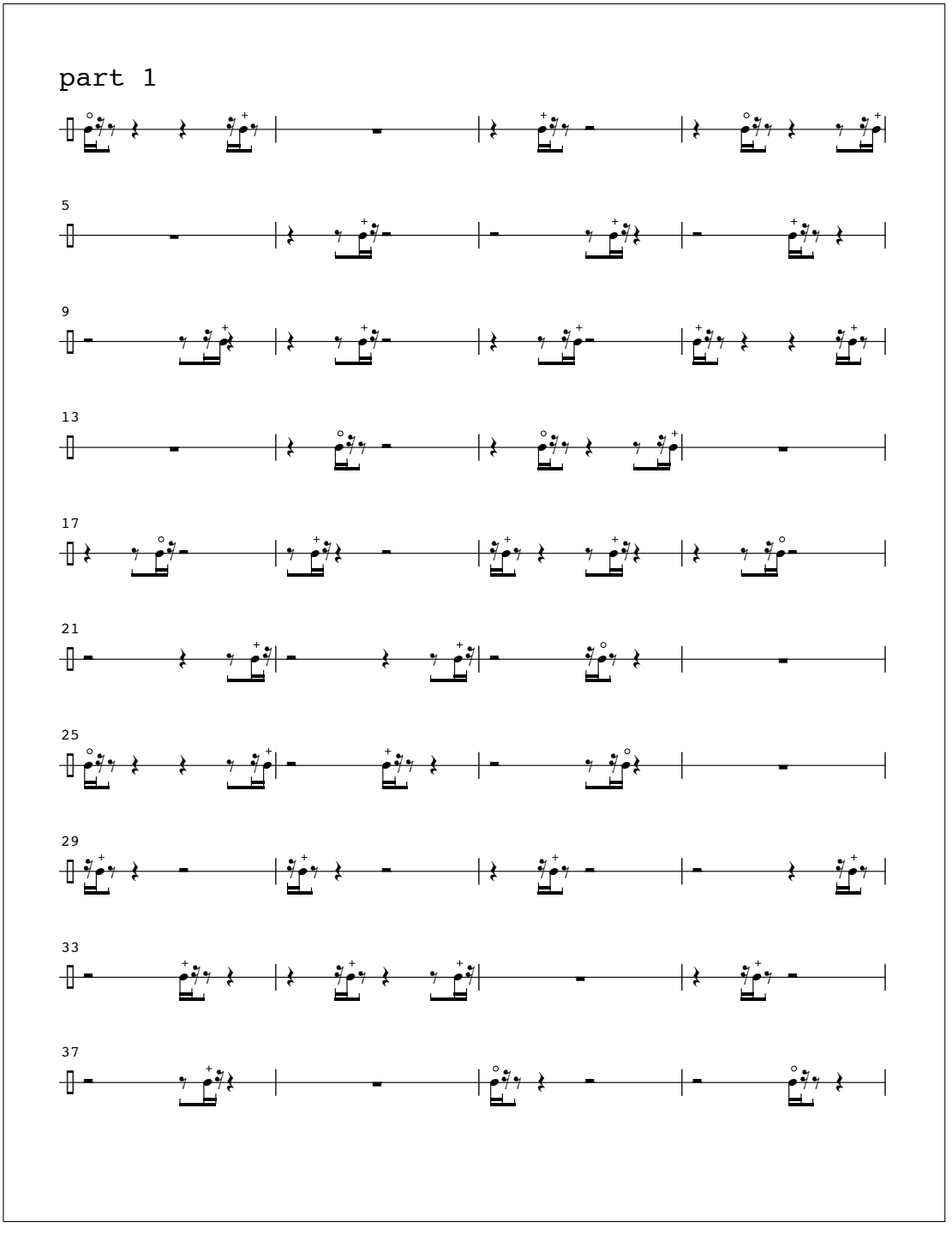
to occur at least twice during the course of the piece with each occurrence separated by at least 3 minutes (preferably more). the drone should sound at least 3 minutes (preferably more) before and after the first and last occurrence.

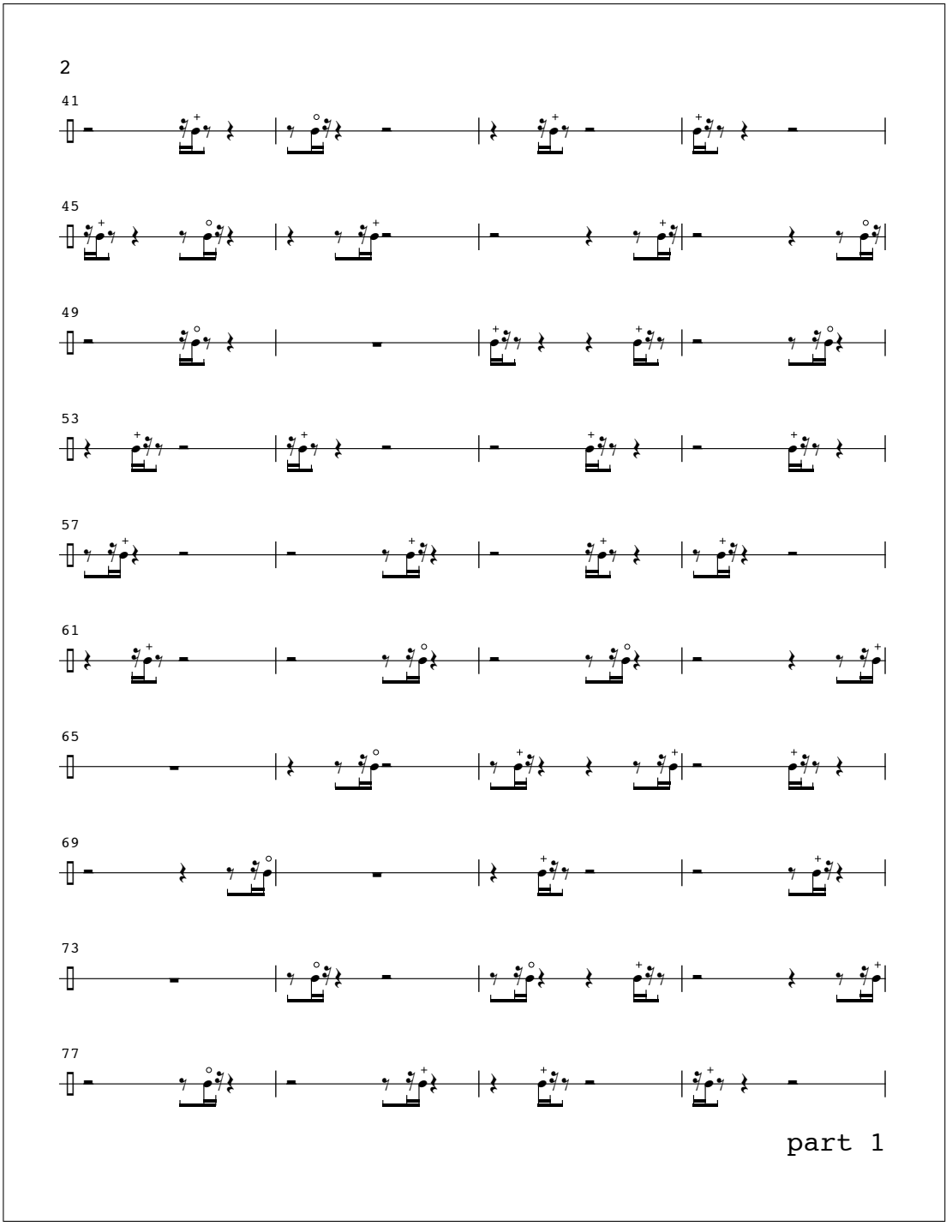
loud. heard equally with, if not slightly above, the drone.

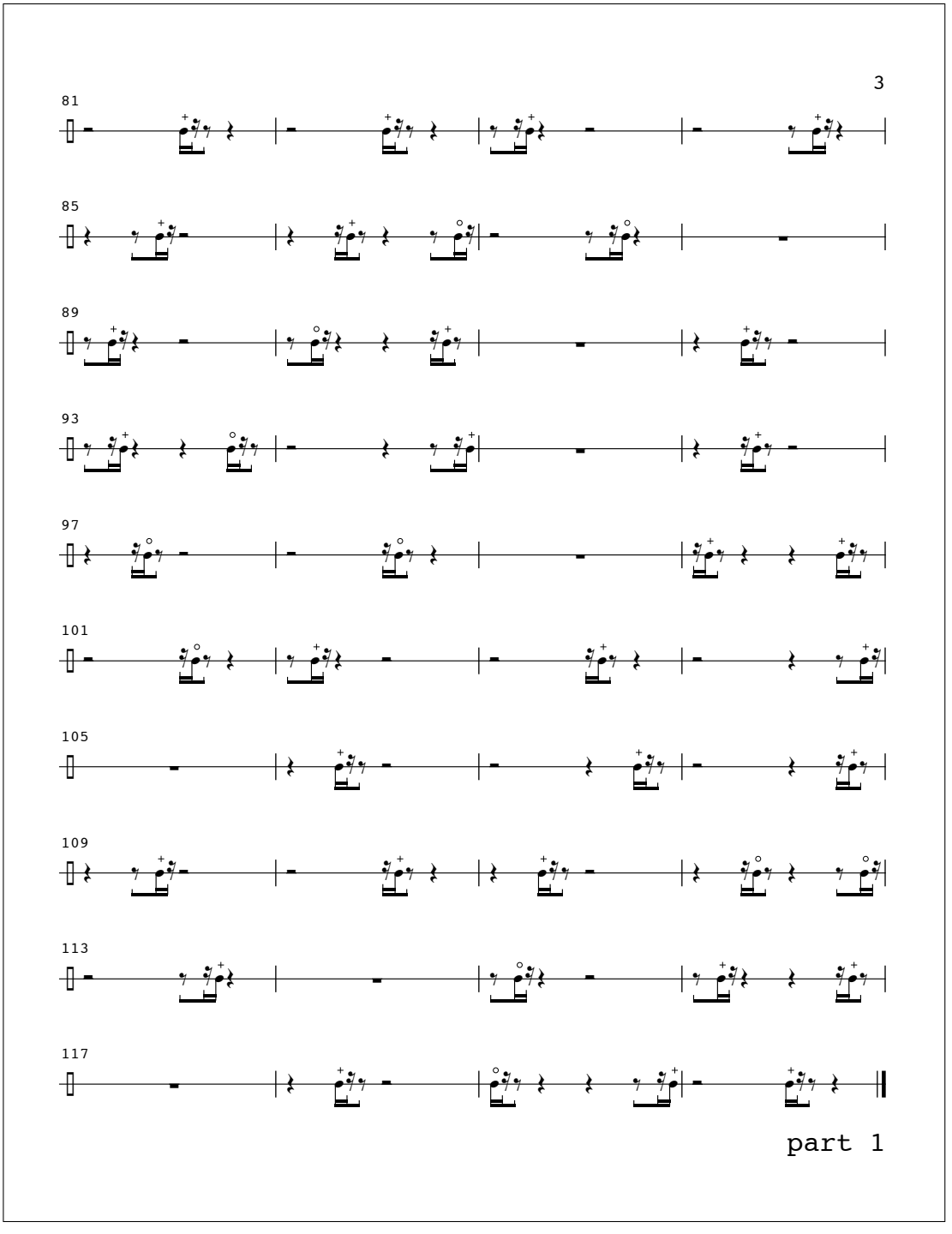
coda (optional):

to occur at least 3 minutes after the last triangle machine occurrence and ending shortly before the drone halts. the coda consists of a 3 minute texture synthesized by a custom-made computer program. the application is designed for multichannel playback through as many speakers as possible distributed throughout the space. the texture should be subtly noticeable; blending with the drone.

-*michael winter* (los angeles; february, 2010)


part 1
5
9
13
17
21
25
29
33
37


2
41
45
49
53
57
61
65
69
73
77
part 1


3
81
85
89
93
97
101
105
109
113
117
part 1