

University of Adelaide
Elder Conservatorium of Music
Faculty of Humanities and Social Sciences

**Declarative Computer Music Programming:
using Prolog to generate rule-based musical counterpoints**

by

Robert Paweł Wolf

Thesis submitted in fulfilment of the requirements
for the degree of

Doctor of Philosophy

Adelaide, August 2014

CONTENTS

Abstract	4
Declaration	5
Acknowledgements	6
List of Figures	7
INTRODUCTION	10
The Problem	10
The Purpose	12
The Approach	15
PART A: PREPARATION AND PROBLEMS	18
1 Literature review	19
1.1 Literature on Prolog computer language	19
1.2 Literature on counterpoint	21
1.3 Literature on computerised systems for automated generation of music	24
1.4 The current methods and problems of systems for automatic composition	29
2 Methodology	32
2.1 The Construction of the Contrapuntal Expert System	33
2.2 Testing in the Live-Performance Context	34
2.3 Genetic Programming and Code Self-Modifications	35
2.4 Development of Artificial Neural Networks for Aesthetic Assessment	36
2.5 Artificial Neural Network as a Fitness Function for Evolution of the Code	37
PART B: EXPERIMENTS AND COUNTERPOINTS	38
3 Construction of the Contrapuntal Expert System in Prolog	39
3.1 Prolog and other software tools used in the research	39
3.2 Encoding musical notation in Prolog (Input and Output to the system)	41
3.3 Construction of the rules for notes, intervals and musical scales in Prolog	47
3.4 Necessary modifications to Prolog search mechanism	60
3.5 Extracting the rules of counterpoint	67
3.6 Translation of the counterpoint rules into Prolog	68
3.7 Rhythm and rhythmic rules	81

3.8 Entire system architecture	93
3.9 Optimisations and code re-factoring	104
4 Live Performance and Practical Application	108
4.1 Manual input of cantus firmus	109
4.2 Automatic input of cantus firmus from MIDI keyboard controller	109
4.3 Automatic input of cantus firmus from acoustic instruments	110
4.4 Selecting a musical scale or mode	111
4.5 Cooperation between the system and a musician during performance	112
4.6 Generating musical ideas	113
5 Code self-modification	114
5.1 Example of code self-modification	114
5.2 Prolog mechanism for code self-modification	117
5.3 Principles of genetic programming in the context of this investigation	118
5.4 Genetic programming through code self-modification	120
5.5 Generating new musical scales through code self-modification	121
5.6 Mutation of general contrapuntal rules	130
5.7 Mutation of the control code	137
6 Objective assessment of counterpoints	138
6.1 Context	138
6.2 Definitions	138
6.3 Assessment criteria	139
6.4 Examples	140
6.5 Calculating score in Prolog	145
7 Subjective assessment	149
7.1 Context	149
7.2 Definition and requirements for a non-deterministic assessment program	149
7.3 Artificial Neural Networks	151
7.4 Subject of experiments	152
7.5 Software tools used in the experiments	153
7.6 Training a single-neuron network	154
7.7 Training a three-neuron network	154

7.8 Training a multi-layer network	154
7.9 Analysis of the initial experiments	155
7.10 Achieving 100% accuracy	155
7.11 Exploring subjectivity	157
7.12 Experimenting with Neural Networks	158
CONCLUSIONS	161
BIBLIOGRAPHY	167
Primary sources on counterpoint, history of music and musical theory	167
Secondary sources on counterpoint, history of music and musical theory	167
Sources on computing techniques and computer languages	168
Primary sources on computerised music	169
Secondary sources on computerised music	169
APPENDICES	174
Appendix A: Computer Code in C++	175
Lilypond Exporter	175
Artificial Neural Network Extension to Prolog	183
Pitch Tracker	194
Appendix B: Computer Code in Prolog	203
Basic / Initial definitions	203
Generic Contrapuntal Rules	206
Main Code of the Contrapuntal Expert System	209
Scale Definitions	226
Code Mirror	234
Code Evolution	237
Objective Score	243
Counterpoint Generation en masse	246
Appendix C: Musical Examples	251

ABSTRACT

**Declarative Computer Music Programming:
Employing unique features of the Prolog programming language
to generate rule-based musical counterpoints.**

This submission for the degree of Doctor of Philosophy at the Elder Conservatorium of Music, University of Adelaide, is presented as a conventional, text-based thesis, supported by computer code and audio files.

The primary purpose of this research investigation in the field of Artificial Intelligence has been to test the capabilities of the declarative programming paradigm to generate musical counterpoints within the framework of a specially created expert system. The project has tested if such a contrapuntal expert system can evolve through a process of mutation of its own code and generate musical counterpoints that do not conform exactly with the original programming. It presents for the first time a music based study of this capacity for code self-modification.

The expert system developed for this project was constructed declaratively, using the Prolog computer language, rather than the more common imperative approach. Although it is a General-Purpose language, Prolog is particularly effective in the construction of Artificial Expert Systems, because its unique declarative programming style allows the programmer to focus on describing the problem rather than describing how to solve the problem. This leaves to the machine the task of finding the solution to the given problem. The problem in this case is how to generate - artificially - simple counterpoints to short melodic phrases drawn from the cantus firmus tradition. As part of the problem solving process the expert system was taken through a series of evolutionary experiments with Artificial Neural Networks used as a fitness function.

DECLARATION

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint award of this degree.

I give consent to this copy of my thesis, when deposited in the University Library, being made available for loan and photocopying, subject to the provision of the Copyright Act 1968.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

Signature: _____

Date: _____

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my appreciation and gratitude to all the people who were involved in this project, for their opinions, suggestions, criticism, feedback and being with me on this journey.

My special thanks go to Professor Charles Bodman Rae, my principal supervisor, for his initial impulse to start this academic journey. Together with Dr. Luke Harrald, they formed a multi-disciplinary supervising team, which was essential to the successful completion of this research project. By having these two experts to guide me through meanders of professional academic activity, I consider myself lucky. While Professor Bodman Rae helped me greatly to express clearly my ideas in musical and musicological contexts, Dr. Luke Harrald supervised the technological aspect of the project and skilfully guided me through two ACMC conferences.

This research was greatly influenced by my “venerable master” Dr John Polglase, who led me through the intricacies of the art and craft of counterpoint. He caused me frequently to reflect on the issues of beauty versus de-humanised machine processing.

Many thanks go to Associate Professor Kimi Coaldrake for overseeing the administrative side of this project and support. Also I would like to thank Mr Stephen Whittington for presenting me with many questions of a philosophical nature and his feedback on my English prose. I also wish to thank Dr. Michelle Picard for making me more aware of the style and conventions of academic writing.

I feel it is appropriate to acknowledge here the scholarship support of the Australian Federal Government through the Australian Postgraduate Award.

And last, but by no means least, I offer my special thanks to my beloved wife, Grażyna, for her support, encouragement, patience and our endless discussions on the subjects of emotion, intellect, soul and their machine equivalents. All this was essential to formulating many of the ideas and computer code presented in this dissertation.

LIST OF FIGURES

Fig. 1: Prolog symbols for encoding musical pitches.....	42
Fig. 2: Scale A-major notated in Prolog.....	43
Fig. 3: Prolog representation of note durations.....	44
Fig. 4: Two methods of specifying notes with durations.....	45
Fig. 5: Three methods of specifying musical rests in Prolog.....	45
Fig. 6: Examples of some intervals written in Prolog.....	46
Fig. 7: Two-part musical example written in Prolog.....	46
Fig. 8: Examples of using the interval relation.....	48
Fig. 9: Definition of the notestep relation.....	49
Fig. 10: Definition of the octave relation.....	50
Fig. 11: Definition of the notevalue relation.....	50
Fig. 12: Definition of the interval relation.....	51
Fig. 13: Comparison with other language, such as Lisp.....	51
Fig. 14: Examples of Prolog relations working in reverse.....	51
Fig. 15: Simple definition of a major mode using the interval relation.....	52
Fig. 16: Sample use of the scale_major relation.....	52
Fig. 17: Definitions of members of minor scale.....	54
Fig. 18: Code model for representing negative relations in the system.....	55
Fig. 19: Prohibited melodic motions in minor mode (first rule).....	56
Fig. 20: Prohibited melodic motions in minor scale (four other rules).....	57
Fig. 21: Major leading note can only be followed by tonic.....	58
Fig. 22: Major sub mediant can only be followed by major leading note.....	58
Fig. 23: Definition representing the fact that all melodic motions are allowed.....	58
Fig. 24: All possible starting and ending notes of melodies in minor scale.....	59
Fig. 25: Sample program for composing 5-note melodies.....	60
Fig. 26: Order of generating melodic results by Prolog.....	63
Fig. 27: Possible definition of the rres relation.....	64
Fig. 28: Possible definition of random_cl relation.....	65
Fig. 29: Sample program for composing 5-note random melodies.....	65
Fig. 30: Definition of sres and melody.....	66
Fig. 31: Sample queries and their results.....	66
Fig. 32: Contrapuntal rule for checking voice cross-over.....	69
Fig. 33: Contrapuntal rule specifying perfect intervals.....	70
Fig. 34: Contrapuntal rules specifying available melodic motion intervals.....	71
Fig. 35: Contrapuntal rule grouping all available melodic motion intervals.....	72
Fig. 36: Contrapuntal rule excluding repeated notes from melodic motion.....	72
Fig. 37: Contrapuntal rule prohibiting leaps in more than one voice.....	73
Fig. 38: Contrapuntal rule prohibiting unison achieved by a leap motion.....	73
Fig. 39: Contrapuntal rule prohibiting more than two repetitions.....	74
Fig. 40: Contrapuntal rule prohibiting parallel perfect intervals.....	74
Fig. 41: Contrapuntal rule defining similar melodic motion in two voices.....	75
Fig. 42: Contrapuntal rule prohibiting virtual parallel perfect intervals.....	76
Fig. 43: Contrapuntal rule prohibiting too many parallel intervals.....	77
Fig. 44: Contrapuntal rule guarding correct resolutions of leap intervals.....	78
Fig. 45: Contrapuntal rule prohibiting more than 2 skips in one direction.....	79
Fig. 46: Contrapuntal rule prohibiting compound dissonances.....	80

Fig. 47: Contrapuntal rule describing dissonances.....	80
Fig. 48: Symbols used to describe rhythmical patterns.....	81
Fig. 49: Contrapuntal rule defining all possible rhythmical pattern symbols.....	82
Fig. 50: Contrapuntal rule defining the rhythmical pattern of the first bar.....	83
Fig. 51: Contrapuntal rule defining the rhythmical pattern of the last bar.....	83
Fig. 52: Example showing typical usage of barstt relation with differential list.....	85
Fig. 53: Bar transition table definition 1.....	85
Fig. 54: Bar transition table definition 2.....	86
Fig. 55: Bar transition table definition 3.....	86
Fig. 56: Bar transition table definitions 4, 5, 6 and 7.....	87
Fig. 57: Bar transition table definitions 8, 9, 10 and 11.....	88
Fig. 58: Bar transition table definitions 12, 13, 15, and 16.....	89
Fig. 59: Typical usage of florid_barring definition.....	90
Fig. 60: First definition of the rhythmical pattern generator.....	91
Fig. 61: The final two definition of the rhythmical pattern generator.....	92
Fig. 62: Initial blueprint of the entire contrapuntal expert system.....	94
Fig. 63: Second attempt – including of the scale parameter.....	94
Fig. 64: Third version – inclusion of vertical relative control.....	95
Fig. 65: Fourth version – inclusion of information about counterpoint.....	95
Fig. 66: The prototypes of all necessary sub-definitions.....	96
Fig. 67: Typical definition of single species.....	97
Fig. 68: Definition of the first two bars of the first species counterpoint.....	99
Fig. 69: Filling the definition with contrapuntal constraints.....	100
Fig. 70: Complete definition of starting point of the first species.....	101
Fig. 71: The definition of recursive processing of the first species.....	102
Fig. 72: Recursion ending definition of the counterpoint_continuation.....	103
Fig. 73: Reduction of the amount of parameters.....	106
Fig. 74: Code re-factoring – separation of declarative parts from imperative.....	107
Fig. 75: Counterpoint generated during live performance on 3rd April 2013.....	112
Fig. 76: Generation of musical ideas.....	113
Fig. 77: Fibonacci series programmed in Prolog.....	115
Fig. 78: Alternative method of definition.....	115
Fig. 79: Self-modifying version of Fibonacci series.....	116
Fig. 80: Operations for code manipulations.....	117
Fig. 81: Typical structure of a file containing definitions of musical modes.....	122
Fig. 82: Preparation steps for generating mutated musical mode.....	123
Fig. 83: Code responsible for creating mutated file.....	124
Fig. 84: Code for altering one note from a scale.....	125
Fig. 85: Code responsible for exporting mutated definitions.....	127
Fig. 86: Sample result generated by mutation of a musical mode.....	129
Fig. 87: Counterpoint generated with mutated mode a-minor.....	130
Fig. 88: Program mutating contrapuntal rules.....	131
Fig. 89: Program for choosing type of mutation.....	131
Fig. 90: Program for removing a prohibitive clause.....	132
Fig. 91: Code for mutating a specific clause.....	133
Fig. 92: Program for mutating numeric values in Prolog definitions.....	135
Fig. 93: Code for conducting mutation experiments.....	136
Fig. 94: Sample result violating the compound dissonance check.....	136

Fig. 95: Counterpoint statistics examples.....	140
Fig. 96: Structure of a typical file with counterpoints.....	142
Fig. 97: Structure of exported counterpoint.....	143
Fig. 98: Pseudo-code.....	146
Fig. 99: Checking parallel imperfect consonances.....	147
Fig. 100: Calculating score for parallel imperfect consonances.....	148
Fig. 101: Cantus firmus used in experiments.....	152
Fig. 102: Sample counterpoint with numeric representation.....	153
Fig. 103: The result obtained from the original training set.....	159
Fig. 104: The result obtained from the randomly re-ordered training set.....	159