Rule-based Music Composition using Cantus Firmus

Daniel Alarcón

Supervisors:
Ian Davies & Dr Shayne Flint

COMP8790: Software Engineering Project
Masters of Computing
Semester 2, 2013
Acknowledgements:

My supervisor, Ian Davies, deserves the greatest thanks. This work, for what it’s worth, would not exist if were not for his expertise (knowledge and keen eye regarding music), feedback, patience and support. I would also like to thank Dr Shayne Flint, his countenance, aplomb and support are highly valued. Some special gratitude is due to Dorien Herremans of the University of Antwerp Operations Research Group (ANTOR) whose papers offered much valuable insight into implementing this software, and who personally provided me with source code and informative discussion. My family and friends also deserve gratitude for their emotional support and company.
Abstract:

This project describes a rule-based approach to Computer Aided Composition (CAC). It represents more than 400 years of contrapuntal theory and an expert composition rule-set. We have undertaken this to improve our understanding on the current state of work in the area and the promise that the future may hold for research in this domain. Music theory in voice leading, which these rules address, has been well-developed for centuries. However, CAC has rarely taken a rule-based approach. Instead the field has been dominated by statistical, probabilistic, random (or “chance”), and mathematical (using fractals, for example) approaches. These approaches are not able to capture human decision-making in the same way that a rule-based method can. This rule-based approach takes a perspective of vocal parts rather than harmonic progression. Harmonic progression will be an emergent part of the rules of voice leading.

Although these rules were originally developed as a teaching tool I believe it is possible and valuable to formalise these rules in the Computer Science context and perform analysis in a methodical, structured way to potentially discover, using formal methods, further improvements and optimisations.
## Contents:

1. Introduction ........................................................................................................... 5
2. Related work .......................................................................................................... 10
3. Software .................................................................................................................. 12
   3.1 Algorithm .......................................................................................................... 12
   3.2 Implementation ................................................................................................. 13
4. Discussion ................................................................................................................ 17
   4.1 Results ................................................................................................................ 17
   4.2 Improvements ...................................................................................................... 20
5. Lessons Learned ..................................................................................................... 21
   5.1 Research ............................................................................................................. 21
   5.2 Software .............................................................................................................. 22
   5.3 Presentation ........................................................................................................ 23
6. Conclusions and Future Work .................................................................................. 24
   6.1 Future Work ........................................................................................................ 24
      6.1.1 Rule Analysis using Logics ............................................................................ 27
   6.2 Conclusion .......................................................................................................... 29
7. References ............................................................................................................... 30
8. Appendix .................................................................................................................. 31
   8.1 Glossary ............................................................................................................. 31
   8.2 Rules ................................................................................................................... 36
   8.3 Algorithm (pseudo-code) .................................................................................. 44
   8.4 Notes ................................................................................................................... 45
      8.4.1 Redundant CF rules discovered .................................................................. 45
      8.4.2 Redundant CP rules discovered ................................................................. 45
1. Introduction

In this report I will discuss the analysis and implementation into software of some of the oldest formal rules ever defined for music.

The rules I refer to are the “species counterpoint” (also known as strict counterpoint) first published by Girolamo Diruta in 1610 in his *Il Transilvana* (however first mentioned, conceptually, by Giovanni Maria Lanfranco, in 1532 in his *Scintille di musica*) from which the more widespread and well-developed technique was derived by Johann Joseph Fux in his *Gradus ad Parnassum* in 1725 [2].

The “species counterpoint” technique (hereafter referred to as the Fuxian method) is a gradual expansion on a single line of notes called a Cantus Firmus. Cantus Firmus is a term from as long ago as the 14th century referring to a “fixed melody” upon which further lines of music, also conforming to rules of the same kind, are superimposed for a full contrapuntal effect [6]. These rules of counterpoint are “time independent”; meaning neither the passage of time, nor any rhythm is applicable to a set of notes satisfying these rules. This set could represent, for example, a musical ornament, a melodic phrase, or the arrangement of movements of a larger piece. Thus, these notes are simply an ordered set.

J.S. Bach employed and may serve as a perfect exemplar, if not in part responsible, for the impact and development of these “principles of counterpoint”, represented as this Fuxian method. This method is, essentially, the invisible foundation of classical western music but has largely been ignored in modern Computer Aided Composition (CAC).
There is a rich history of algorithmic approaches to music generation; for example, Mozart was known to have a Musikalisches Würfelspiel (German for "musical dice game") where he would use a dice to help decide how to put together (generate) a Minuet [10]. Additionally, certain rule-based approaches have been tried by Athanasius Kircher in 1650 where a poem could be translated into a piece of music using a cypher [14].

The first computer-generated music was the Illiac Suite written in 1956 [12] using the ILLIAC (Illinois Automatic Computer) supercomputers by Lejaren Hiller and Leonard Isaacson using some version of the Fuxian Method [15].

More recently, Iannis Xenakis published his Formalized Music: Thought and Mathematics in Music in 1963. Here he expounds upon his stochastic mathematical functions for music composition [17].

Karlheinz Stockhausen, in 1970 developed a formulaic approach to composition, where certain very informal algorithms are applied to the pitch, dynamics, duration, timbre and tempo of any part of a contrapuntal piece [18].

In 1977, Pierre Boulez established the IRCAM (Institut de Recherche et Coordination Acoustique/Musique), where programmable music was developed. This was completely realised in the mid-1980s with Max, an authoring system for interactive CAC. The primary principles applied for the method behind Max are spectral music and fast Fourier transformations [16] [20]. These involve compositional decisions that are informed by sonographic representations and mathematical analysis of sound spectra [19].
The study of the last several hundred years of western music reveals enough consistency that rules can be formulated to represent common practice. Although all composers “break” these rules from time to time, their development and usage suggests the possibility that these rules represent some degree of universal aesthetic. Athanasius Kircher believed that “the harmony of music reflected the proportions of the universe” and this inspired many of his claims and efforts – he wrote an 1125-page treatise on such concepts [13].

A listener will normally expect certain changes of tone at certain points in a piece – where denial of these expectations can have a clashing or unsatisfying effect. The rules were not created arbitrarily by musicians, but rather, were derived from the musical experience itself and deductive/inductive reasoning. In general, the importance of the Cantus Firmus is to provide balance, poise and a framework for elaboration into a piece. This is analogous to the use of the golden ratio in other creative arts – it is not a rule to follow naively, but a universal observation, departure from which will have meaning. For example, the golden ratio is a balance point in a space. If an artist wishes to have the focus of attention in the space away from this point it will be for some aesthetic purpose that achieves some particular desired effect because of the departure from the golden ratio. Similarly, departure from these Fuxian rules may be employed deliberately by a composer for a particular effect.

This report describes implementing first species counterpoint. First species is note-against-note counterpoint where only one note is ever
placed above or below another note and parts do not cross. The
counterpoint is built up to 3 voices but additional voices are possible.

There is great potential for formal Computer Science analysis in this rule-
based approach, such as: learning of new rules, using Data
Mining/Machine Learning techniques on existing music to elaborate the
rules, and also for rule-elimination (of redundant rules) using logical
inferences by means of formal methods (c.f. Section 6.1.1). Optimisation is
also likely to be a result of this analysis. This will be a primary focus of
discussion.

The code was developed in Java from an initial collection of 75 rules using
a Constrained Depth-first Search Algorithm implementing Backtracking.

There are many challenges presented. There are a very large number of
possible valid counterpoints that can be generated. Assuming a two
octave compass and a maximum length of 16 notes, the search space is
15\textsuperscript{16} possibilities for each voice in the absence of limits imposed by rules. It
is unknown precisely how many “valid” solutions there are because it will
depend on the precise rule set used. Choosing an appropriate search
heuristic to use in traversing the search space is also a challenge.

On the following page is an illustrative example from J.S. Bach’s \textit{Well
Tempered Clavier, 1, Prelude XXIV}. The first is the piece as composed by
Bach. The second is the same piece where the Fuxian Cantus Firmus and
related Fuxian counterpoint lines have been left behind after removing
the elaboration (“frills”) leaving the “bones” or underlying architecture of
the piece exposed.[1]
Figure 1: Bach WTC 1 Prelude XXIV
Figure 2: Bach WTC 1 Prelude XXIV; Fuxian parts only
2. Related Work

In March 2013 Dorien Herremans, of ANTOR, developed FuX, an App which composes 5th Species Fuxian counterpoint [3] [23]. However, their implementation uses fewer rules, where some of the rules held in common are less restrictive. Their search heuristic is a Variable Search Neighbourhood algorithm [3]. In general, such an algorithm “explores distant neighbourhoods of the current incumbent solution, and moves from there to a new one if and only if an improvement was made. Local search method is applied repeatedly to get from solutions in neighbourhood to local optima” [22]. In the ANTOR implementation their algorithm is one “that starts from a randomly generated fragment and gradually improves this solution by changing one or two notes at a time” [3]. Their method of defining a “better” solution is derived from adherence to the Fuxian rules.

Similar work has also been performed in Spain, however their implementation involves probabilistic algorithms [5]. We have decided, in this work, that using probabilistic decisions unduly hinders the capacity of the software to search the space to explore the consequences of the application of the rule-set. We prefer to fully explore the ramifications of these rules without the a priori biases imposed by probability distributions.

Some aspects of the Fuxian method have also been implemented in a real-time composition software by Anders and Miranda [21]. Their research extends their previously developed music constraint system called
“Strasheela” with real-time capabilities. Strasheela is based on the multi-paradigm programming language Oz. Their work exploits paradigms of declarative concurrency (concurrent constraint programming plus first-class procedures) and constraint programming based on computation spaces.

In modern CAC, machine learning is not often implemented for the sake of rule-learning or rule-improvement; and in particular, not in the context of the Fuxian method.

Furthermore, the rule-based approach in general, in modern CAC, has been largely neglected, instead opting for statistical, probabilistic, random (or “chance”), and mathematical (using fractals, for example) approaches [7] [8] [9] [11].
3. Software

3.1 Algorithm

The algorithm used in this implementation emphasises unbiased random choice from a set of possibilities defined by the rule-set. This is to say, the rule-set is not subjected to biases introduced from probability distributions. A note is chosen from the set of possible tones allowed by the rule-set at each step entirely at random; not sequentially or by any other heuristic. This allows richer exploration of the domain permitted by the rules.

The Cantus Firmus (CF) is always generated first with the initial tone given. Following this there will be a set of available tones – this is determined by a subset of the first 27 of the existing 75 rules. A tone from those permitted by the rules is selected at random. Application of each rule at each step is in the context of all choices made up to that time. This continues until a termination condition is reached. If there are no options to proceed, the search backtracks and resumes on another branch. Either there are no more possible tones to follow the last, or the CF has reached a maximum (16) (where the minimum is 8) in this implementation, or when the line generated thus far qualifies as a valid line as per the rules. The limit of between 8 to 16 notes is not arbitrary but is one of the rules of the Fuxian method. To test that the line qualified as valid it is subjected to the remaining rules from the 27. Pseudo-code for this algorithm can be seen in Section 8.3.
This strategy is naïve in the sense that there exist logic shortcuts embedded in the rules, and this will be discussed in Section 4.2.

Once there is a valid CF generated (which takes as little as a few milliseconds and as much as 50ms), a counterpoint (CP) (a second voice or part) is sought. The process of finding a valid CP is exactly the same as that for the CF, except that the remaining 48 rules are used – these rules are different because now “harmony” must be taken into consideration. The application of these rules requires unbounded look-behind and even some look-ahead of the CF; this is repeated once more for the 3rd and final line of CP for this implementation. This also has some inefficiencies which are also addressed in Section 4.2 as the CF generation improvements. Because the available possibilities are deterministic i.e. form a tree, the algorithm has been classed a Constrained Depth-first Search with Backtracking.

3.2 Implementation

The implementation itself was coded in Java, using native MIDI tools for output. There are also detailed logs, including random number seeds and other results from execution for use in testing and debugging. Care in managing random number seeds also facilitates repeatability.

The project is a proof-of-concept, and while what has been implemented is somewhat rudimentary, it nevertheless provides useable and interesting results. The focus is ultimately the rule-based approach and the opportunity this affords formal analysis and exploration of the domain.

Music is being modelled, theoretically, using the equivalent of simple integer arrays. Each line or “Voice” is one array. Each element of each
array is an offset from the “key” in which the piece will be heard using semitones as the unit of measurement.

Each line has a wrapper governing its generation. There is a class to represent a generic line (“Voice”) containing the logic to which all Fuxian lines must adhere. There is then a class for the CF line (of which there is only ever one instance) and one instance of a CP class is used for each CP line. The CF class has the logic needed to track and control generation of a valid Cantus Firmus. The CP class has the logic needed to track and control generation of valid counterpoints over the CF and other, already generated CPs. This is all managed by a “God class” (essentially representing the “Score”) which contains these lines and manages the outputs (the MIDI sound and file, and the logging data).

Due to time constraints only the major diatonic scale was considered, and only 1st Species in three parts has been implemented. Additional rules remain to be written. This project attempts to translate rules expressed in natural language into what is, essentially, a Java-codified predicate logic. In some cases rules were excluded due to ambiguity, requiring further research (perhaps using formal predicate logic calculi c.f. Section 6.1.1), or because they proved redundant. However, the design is such that, the implementation of future tasks can be done without massive restructuring.

Figure 3 is a UML class diagram conceptually representing the system to help illustrate the above explanation. The class names are not precisely those used in the implemented code, and some of the classes shown exist
only implicitly in the implementation – e.g. “Scale System” exists implicitly in the allowed values of the offsets which are defined explicitly in “Voice”.

The code has been written in such a way that many of the central rules are parameterised. Thus, in its current form, some rules can be modified by the user, as specified in the provided parameters. Other non-rule-related parameters also exist, such as one providing a user-defined random...
number seed and one to specify the number of counterpoints to provide to the Cantus Firmus.

This verification process consisted purely of manual inspection and submission to the keen eye of a qualified musician. Since all the rules are known and the output is clear, it is not restrictive to check correctness through inspection.
4. Discussion

4.1 Results

The implementation was, largely, a success. The scope was fulfilled; where everything described in Sections 3.1 & 3.2 represented the expected scope. The requirements of rule analysis and codification, of extensibility, parameterisation, and of development of a fair algorithm were fulfilled. The project implementation serves as the proof-of-concept that it was intended to be. It may also serve as a stage for the further formal analysis techniques (c.f. Section 6.1.1) and the platform for the powerful tool envisioned for the future (c.f. Section 6.1 ix).

Most importantly, the software successfully outputs “pleasant” sounding music as per the intention of the Fuxian method. This is both audible immediately using the software and also logged as both a MIDI file and as raw text data.

All the Fuxian rules were implemented within the scope of the diatonic scale. However, there was a notable issue encountered in the codification of the rules. The rule forbidding parallel 5th and 8th has been excluded from the final generation process and thus outputs show instances of this characteristic (i.e. they may contain parallel 5th or 8th) and thus are not entirely compliant with the Fuxian rules. This is precisely because of the combination of rules causing a condition of unsatisfiability. That is to say, the addition of this particular rule causes an immediate deadlock, where the mutual/simultaneous satisfaction of the all of the rules becomes impossible at a given time and output is never received. Although it has
not been formally shown that the generator would not eventually produce
a satisfying, valid CF/CP, there is a very clear impression of inability for the
generator to produce the expected output.

The most likely culprit for this condition is the collection of rules from
several sources. The most informative rule was chosen when two rules
addressed the same characteristic – i.e. the strictest rule-set prevailed.
While all the rules derived from the same origin, and thus all valid in
Fuxian terms, they are clearly in a state of mutual disagreement (more
precisely, “unsatisfiability”). Rectification of this would require analysis
like that which is discussed in Section 6.1.1.

On the following page, there are 3 examples of well-formed (except for
parallel 5\textsuperscript{th}s or 8\textsuperscript{th}s, as mentioned) CF with 2 CPs generated by this
implementation.

The music shown in Figure 3 can be listened to at: \url{http://goo.gl/GsGMxy}

The music shown in Figure 4 can be listened to at: \url{http://goo.gl/vXxlgI}

The music shown in Figure 5 can be listened to at: \url{http://goo.gl/n7WPUJ}
Figure 2: Generated using seed 101645655517

Figure 5: Generated using seed 195299903456534

Figure 6: Generated using seed 1954068652042745
4.2 Improvements

The algorithm itself is fully implemented and operational. However it is yet somewhat naïve in some ways. This is not due to lack of understanding but rather lack of time to analyse as needed.

Clearly, there is enormous capacity for learning while backtracking. Learning, in this case, would involve recording choices which lead to failed CF or CP generation, compiling them into a library, and then eliminating them preemptively in future. This further leads to the possibility of learning new rules entirely, if these choices present patterns which can then be incorporated into the rule-set – i.e. the creation of an extensive and powerful library is foreseeable. This is interesting because it may offer further insight into music theory and may serve to expand the Fuxian method in general. Derivation of more powerful or parsimonious rules would result in faster generation and would expose redundancies.

Currently, backtracking is limited to a particular line, rather than the entire piece. This means that whenever any line fails to produce a valid CP (a CF is always found) then the piece is abandoned entirely. Clearly, an improvement would be to backtrack into preceding lines to guarantee a valid termination and fully explore the space.
5. Lessons Learned

This is my first attempt at something resembling research. And following are some improvements I would like to make in the future:

5.1 Research:

I would prefer to have gathered more “Related Work”. While there were not many papers to be found addressing this particular topic, there may be more loosely related, but yet interesting work among the academic literature. I had not fully realised the value of this process. Additionally, from the beginning I would have like to use the “Related Work” sections from the papers that I did find to send me off in the various directions to which the subject extends.

I would like, in future, to perform more independent experiments. I have implemented the Fuxian rules in a way I thought would suit it. However, it would have been better to perform more experiments to find the best way to explore this project’s domain. I cite my time constraints as the culprit to this particular attempt’s shortcoming.

Many of the “best” ideas, such as formalising the Fuxian rules as propositional logic and applying well-founded logical calculi to them to find interesting properties, and the real-time elaboration using a diverse set of possible interfaces, came quite late in the project. In future, I would like to take more time, early in the project, to think about such “possible innovations”.
Finally, though it seems common sense, I learned how valuable an interested, intellectually competent, dedicated and knowledgeable supervisor (or supervision team) is. While the project was perfectly in line with my heart’s desire, there was an enormous amount that I did not know in the domain (musically) and without the supervisor’s ideas, feedback and counsel this project would not have been viable.

5.2 Software:

I implemented the data structures of the musical notes as “offsets to the middle-C” (which can be easily transposed into any key) in “semitones”. However, other data structures may have been better-suited to this application. I should have conceived, or sought more alternatives and applied a process to select a “best”. Again, I cite time constraints as the culprit for this shortcoming.

Since this project was conducted under the title of “Software Engineering” it may have been remiss of me to neglect modelling methods and even some kind of formal Requirements Specification earlier in the project. I thought because of the discreteness of the task, the single-worker structure of development, and my prior industry and private experience, that this would be OK. However, I now feel that some formal modelling and analysis of these models in the early stages of the project may have been valuable.

In reference to the above point, I feel, however, that my implementation worked out rather well. So, my experience and understanding of OO programming, and mental modelling of a problem is not as badly-founded as may be imagined.
I would like in future, to subject my code to further peer-review. I know this is valuable, and I think that this project may have benefitted from “more eyes” scrutinising the code. Again, I feel that the strict time-limits are, at least in part, to blame for this being more-or-less impossible.

5.3 Presentation:

Because my supervisor is (highly) music-literate, and I have a fairly strong background in music, I found it difficult to frame the discussion in a way that those with no music background can take interest and follow the explanations. I should have taken more time to think this through beforehand.

Something I had already learned, but have now had emphasised, perhaps due to the academic (rather than business-like, or creative writing) context of this report, was the benefit of a careful organisation of ideas before committal to paper. I normally prefer to have all the ideas in my head and simply “dump” this to a document. However, for this kind of technical work I now feel that more of “write-as-I-go” approach may have been more effective. It did “come together” in the end, however the effort for this to occur was greater than necessary.
6. Conclusions and Future Work

6.1 Future Work

As detailed, there are many improvements available for the algorithm on a theoretical level. However, for the broader problem of CF/CP generation via Fuxian method, and for the rigorous analysis of formal methods and machine learning, there is much left to expand upon in this project. Namely:

i. The implemented rules must be checked more rigorously for accuracy. There are many sources of rules which may rightly be called Fuxian rules. This can lead to a condition of contradiction, dilution, and ultimately misrepresentation of rules as code.

ii. Scales other than the “standard” (major) diatonic scale are needed to be implemented. In particular, generation in the minor scales must be addressed. This would immediately double the space of possible specimens. Currently this implementation of the Fuxian rules into Java code only generates counterpoint in the major scales. Additionally, there exist a number of musical scales besides the more prevalent diatonic. For example: quarter tone scale, Phrygian scale, Persian scale, Mixolydian mode, Dorian scale, blues scale. These scales may indeed have their own rule-sets which differ from the Fuxian rules, however exploration of this issue is beyond the scope of this work. Note, however, that the architecture of the software will allow for relatively easy transition to other scales and implementation of alternate rules.
iii. The remaining 4 Species must be analysed, with rules appropriately gleaned and then implemented.

iv. The option of additional voices being included is lacking. This implementation has a maximum of 3 voices. There is not necessarily a logical limit on the number of voices which may be super-imposed on the Cantus Firmus. For example, William Byrd wrote in 6 and even 8-part counterpoint at times [24], and 4 or 5-part counterpoint was common for J.S. Bach. Indeed, the audible frequencies and the possibility of increasing unsatisfiability of the rules would be the only limiting factor. However, the details of this issue cannot be known without a more lengthy and deep investigation.

v. Modulation (i.e. dynamic key changing during generation) is very large and complex component and will be needed in the future. This is actually not Fuxian, however would enhance the power of the tool.

vi. Currently, the CF is placed as the base-most voice - this is often the case in practice, however the generator must be made to account for the other possibilities. These other possibilities and their implications are detailed and well-understood.

vii. The language used by the writers of the rules includes ambiguities and effects the output. Indeed, these rules are in “natural language” used for instruction of students by teachers. So, these are difficult to express as algorithms much as the rules of language grammar are. This will need further analysis and rectification.
viii. A GUI for exploration of the output with power to manipulate elaboration and variation. This would entail a very powerful musical composition and exploration tool.

ix. Fuxian method allows for reverse-engineering of complete pieces (any Bach piece for example) into the corresponding Cantus Firmi. Music students are routinely given the task of analysing music in terms of these species counterpoint rules. Typically, a work can be stripped of notes that are merely ancillary or decorative to reveal the underlying species counterpoint on which it is based. While these notes are essential to the mood and feel of the piece, they are irrelevant to its structure. This raises the possibility that algorithms could be devised to do the opposite, that is, elaborate species counterpoint into a finished work. An apt analogy is the application of transformation algorithms applied to images – for example, any landscape or portrait can be transformed by an algorithm to look like a Van Gogh painting, or have miscellaneous rainbow effects, swirls etc. (effects which may be familiar from products such as PhotoShop) applied to them at the whim or design of a user. The possible computer-human interface implementations are diverse. Gestures or vocalisations could direct elaboration algorithms to operate on an underlying counterpoint to create completed music – possibly in real-time. This rule-based approach to voice leading would create music that “makes sense” whereas mathematical algorithms suffer from sounding “mathematical” i.e. something “not from the human soul”. While this description is not entirely academic, the human experience of music is not yet formalised to a point where such an academic discussion could
reasonably take place. *Figure 1 & Figure 2* showcasing J.S. Bach serve to illustrate the relationship between the Fuxian CF/CP and a complete piece of music – “First we see an instance of alternating suspensions with characteristic imitation at the lower fourth. In this progression – a frequently used idiom of the Baroque period – the imitative pattern implies a harmonic background of ascending fifths... these fifths are filled in with passing progressions resembling third species. The sevenths in the bass, of course, represent inversions of ascending seconds caused by the need to preserve a consistent bass register”[1].

With this additional work, this project would constitute a very powerful tool. This is not to detract from the creativity of a composer however. This work is an “idea generator” at its core; and that is its intended, and possibly most useful, application. There is also much potential for user interaction and real-time customisability. For example, a user could create new rules as their personal aesthetic sense sees fit, tweak any rule, or switch any rules on or off, but this would be minor compared to the ambitious goals of permitting full real-time elaboration (as described in ix).

### 6.1.1 Rule Analysis using Logics

The Fuxian rules may be converted, using deterministic rules, into propositions and framed in first-order logic. For example, at any given point of line generation one could express the rules governing exploration of possible notes in set-builder notation of the form:

\[
\{ x \mid x \in \text{<all notes>} \land \text{rule1(x)} \land \text{rule2(x)}\ldots \text{ruleN(x)} \},
\]
and the line could be expressed similarly. A given rule could be broken down and expressed similarly in propositional logic, for example, the rule:

‘Leap of >4 semitones shall return by step-wise motion’,

could be expressed as:

\[ \neg \text{LEAPOF}0 \lor \neg \text{LEAPOF}1 \lor \neg \text{LEAPOF}2 \lor \neg \text{LEAPOF}3 \lor (\text{NEXTTONE}1 \lor \text{NEXTTONE}2). \]

There are a limited number of similar propositions to define all the rules, and this results in much overlap of aspects addressed between the rules. This then could be subjected to formal logical analysis. Application of logics other than propositional or first-order logic may be possible, however, clearly, propositional and first-order logic will be useful in this context. Inferences could be made on the rule-set as a whole using well-established logic theory tools, such as the array of calculi, usage of programming languages (e.g. Prolog), and SAT tests.

Such analyses would facilitate, for example, the categorisation of rules as redundant, and finding rule simplifications.

In the case of user-defined rules, this technique could also be used to determine “satisfiability”; and thus rules which create a condition whereby lines of music are impossible could be denied from being permitted into the model. When a rule has been denied, such analyses could also generate an output which gives the nearest rule or rule-set to that which has been defined which is, in fact, logically satisfiable.
6.2 Conclusions

This work has shown that this endeavour is entirely possible and that the potential power that formal analysis could add to it would be perhaps even a breakthrough for the Fuxian method. This, in itself, would be exciting for CAC and, in particular, composers. The completed form of this work, which time has hindered from achieving, would act as a useful tool for composers in, for example; rapidly filling in a section, or getting an interesting starting point, or for a suggester of next possible and likely-to-be-good notes, or even as the powerful full elaboration tool suggested in Future Work (c.f. Section 6.1 ix). The totally developed tool envisioned has not yet been fully completed anywhere; at least not outside the minds of geniuses like J.S. Bach.
7. References

[23] ANTOR http://antor.ua.ac.be/FuX on 25/10/2013
8. Appendix

8.1 Glossary

i. *Arrangement* is adaptation of a previously written musical composition for presentation. It may include redevelopment to almost any feature of the piece

ii. *Ascending motion* refers to successive tones playing in increasing pitch

iii. *Bar* is a segment of time defined by a given number of beats

iv. *Cantus Firmus* (CF) (Latin for "fixed song") is a pre-existing melody forming the basis of a polyphonic composition

v. *Chord* is any harmonic set of three or more notes that is heard as if sounding simultaneously.

vi. *Chromatic Progression* refers to tones heard in melodic succession where the difference is precisely one semitone

vii. *Consonance* is a harmony, chord, or interval considered stable

viii. *Contrary motion* is when any 2 lines move in opposite directions

ix. *Counterpoint*, in this context, is a voice heard as played against the Cantus Firmus

x. *Descending motion* refers to successive tones playing in decreasing pitch
xi. *Diatonic scale* is an eight-note musical scale composed of seven pitches and a repeated octave. The pitches increase from the tonic in the pattern: two semitones, two semitones, one semitone, two semitones, two semitones, two semitones, one semitone (the last being the repeat of the tonic at an offset of one octave)

xii. *Dissonance* is a harmony, chord, or interval considered unstable

xiii. *Doubling* is where the same tone (offset by 0 or more octaves) is played in 2 different voices

xiv. *Embellishments* are musical flourishes that are not necessary to carry the overall line of the melody (or harmony), but serve instead to decorate that line

xv. *Harmony* refers to the relationship between notes heard simultaneously

xvi. *Imperfect Chord* refers to an incomplete chord. The chord is comprised of 3 distinct tones. When an imperfect chord is heard one may only be hearing 2 distinct tones

xvii. *Interval* is the space between two notes

xviii. *Key* refers to a particular tonic note and chord tuple

xix. *Leading Tone* refers to the tone found one semitone below the tonic of a scale

xx. *Leap* is melodic motion when not a step

xxi. *Line* is a single line of music, that is, disregarding harmony

xxii. *Measure (see Bar)*
xxiii. *Melody* refers to the relationship between notes heard in succession

xxiv. *Modulation* refers to the key-changes within a piece

xxv. *Movement* is a self-contained part of a musical composition

xxvi. *Note* refers to a musical note

xxvii. *Oblique motion* is where a voice remains on the same pitch while a second voice changes

xxviii. *Octave* is an interval of 12 semitones

xxix. *Ornament (see Embellishment)*

xxx. *Parallel motion* is where lines move in exactly the same intervals in the same direction

xxx. *Parallel* $x^{th}$s where ‘$x$’ is any integer refers to the case where consecutively heard sets of notes each contain an interval of ‘$x$’ within them

xxxii. *Part (see Line)*

xxxiii. *Scale* is a set of notes ordered by pitch

xxxiv. *Score* in the context of this work refers to the combination of all parts

xxxv. *Step* is melodic motion from one note in the scale to one directly “above” or “below”

xxxvi. *Similar motion* is where two lines move in the same direction

xxxvii. *Tempo* refers to the speed or pace of a given piece
xxxviii. *Tone* is the same as a musical note

xxxix. *Tonic* is the first scale degree of a diatonic scale

xl. *Tripling* is where the same tone (offset by 0 or more octaves) is played in 3 different voices

xli. *Unison* occurs when 2 different voices play the exact same tone without any offset

xlii. *Voice (see Line)*

Fux’s 5 Counterpoint Species (in increasing complexity): (all from [2])

- 1st Species is "note-against-note"

![1st Species example](image)

- 2nd Species is "2-against-one"

![2nd Species example](image)

- 3rd Species is "4, 3 or 6 notes against one"

![3rd Species example](image)
• 4th Species is "suspensions (tones held across harmonic changes)"

• 5th Species is a combination of the other 4 (also known as "florid" counterpoint)
8.2 Rules

The 75 Rules of the Fuxian Method [1] [3] [4] [6]:

**CF: (28)**

Length:

- Shall be $7 < CF < 17$

Duration:

- Shall consist of whole notes

Repetition:

- Notes shall not be repeated immediately
- A single note shall appear <4 times
- Lowest note can recur (if lowest is also first, then must recur at end)

Permitted Adjacent:

- Only consonant shall be allowed; $0,1,2,3,4,5,7,8,9,12,15,16$
- When ascending minor sixth is used; shall be followed with descending motion
- Max interval; 12 semitones

Motion:

- Number of leaps shall be between 2 and 4
- Leap of >4 semitones shall be followed by step-wise motion
- Direction shall change after leap >4
- Direction shall change >2 times
- <3 leaps of >4 semitones shall be allowed
- Only 2 consecutive leaps shall be allowed - the 2nd shall be smaller than the 1st and interval between 1st and 3rd note shall be consonant
- The interval between the start and end note of motion in a given direction shall be consonant
- Ascending and descending motion shall occur equally within a margin of 30%
- Interval of a 7th in 3 notes shall not be allowed
- Interval of tritone (e.g. ascending F-A-B natural) shall not be allowed
- Step-wise motion sequences shall be <6
- In the "middle" 3rd, 6th, 10th shall occur more often than other intervals (5th & 8th shall not occur as often as imperfect consonances)

**Patterns/Sequences:**
• Are not permitted (when of more than 2 note) - defined as any set of adjacent notes where the intervals are equal to that of another set of adjacent notes

Climax:

- Shall have a single "highest note" (can be lowest if CP is higher)
- Shall be melodically consonant with tonic
- Shall occur in the "middle" on a strong beat - where the "middle" is the sector of the line where 30% of the note durations have already been played and 30% have yet to be played

Beginning:

- First note shall be tonic (if CP-line can be dominant)

End:

- Last note shall be tonic and in same octave as first note
- Second last note shall be leading tone for CP and the supertonic for CF. If minor, 7th shall be raised by half tone
- Second last note shall not follow a leap larger than a 3rd

**CP (2+line interaction): (47)**

Permitted Interval:

- Only consonant shall be allowed; 0,3,4,7,8,9,12,15,16
• Dissonance between ANY 2 parts shall not be allowed; 1,2,10,11+(augmented or diminished intervals)+5(4th)(sometimes)

• Maximum interval between voices; 16 (except climax)

• Unison (0) shall be allowed only at beginning or end

• 3rd,6th,10th shall occur more often than any other interval

• <4 successive parallel 3rd,6th,10th

• Parallel 5th or 8th between 2 parts shall not be allowed

• Compound intervals (i.e. >octave) enable succession of 5th & 8th by contrary motion and shall not be allowed

• Tritone dissonance (augmented 4th or diminished 5th) shall occur only between upper voices of a diminished (6/3) chord

• (6/4) chord (i.e. a 6th and 4th above a tone - 2nd inversion of chord) shall not occur

• Interval between outer voices (3P CP) shall be <= 2 octaves

• When outer voices (3P CP) interval is an >= octave then the middle voice shall be a smaller interval to the top than the bottom

• Upper voices (3P CP) shall be <octave interval
- Lower voices (3P CP) shall be permitted to $\geq$ octave interval
- Outer voices (3P CP) shall form a perfect 2P CP (excluding interval maximum)
- Open positions shall occur more often than closed (i.e. not as close as possible)
- <4 consecutive intervals of the same kind between any 2 parts shall not be permitted

**Motion:**

- All intervals of 0,5,7,12 (perfect) shall be approached by contrary or oblique motion
- All types of motion shall be used
- Motion in parallel 4th shall not be allowed
- Contrary motion shall occur more often than any other type of motion and steps shall not be equal
- Unison to octave shall not be permitted
- Leaps >6 shall not be permitted in the same direction
- Inner voice (3P CP) and an outer voice shall be permitted to approach perfect consonance in similar motion if the remaining outer voice moves in contrary motion, and if the outer voice is the top voice then the top voice shall not move by leap
- Upper voices (3P CP) shall be permitted to move in contrary motion less than 1/4 of the time

- Parallel motion of all 3 parts (3P CP) shall be permitted only in a series of (6/3) chords - the 6th shall be in the top voice (otherwise parallel 5ths will result)

**Repetition:**

- Shall be restricted to <3 adjacent notes

- Shall be restricted to <3 in line

**Climax:**

- Shall not coincide

**Beginning:**

- First note shall be 0,8th,5th above CF

**End:**

- Last note shall be unison or octave

- Second last measure shall contain leading tone and the 2nd degree to the scale (i.e. the 2 tones adjacent to the tonic) (CF will contain one and CP the other) - If one of the parts is in minor mode the leading tone shall be raised at the cadence. If the leading tone is preceded by the 6th degree of the scale then it must also be raised to avoid melodic intervals of augmented 2nd. The leading tone (raised in minor) and the 2nd step of the scale must
(when adding a 3rd voice) be a complete sonority of 3 tones and, be consonant (diminished (6/3) is considered consonant) - usually when the lowest part is the CF

- When using leading tone in minor, the 7th step in the preceding bar shall not occur c.f. Cross Relation

**Overlap:**

- A line shall not have a note higher than the preceding note of a higher line
- A line shall not have a note lower than the preceding note of a lower line

**Crossing:**

- A line shall not have a note higher than a simultaneous note of a higher line
- A line shall not have a note lower than a simultaneous note of a lower line

**Cross Relation:**

- A chromatic progression shall not occur across any 2 voices

**Method of Joining:**

- Build from the bass, upwards.

**Incomplete Chords (Doubling) (3P CP):**
Doubling shall only occur on imperfect chords - and of the bottom tone

Doubling shall occur more often between the bottom and middle parts than otherwise

<3 in succession shall be permitted

1 tone tripled shall be permitted only in the first or last measure

5th with one tone being doubled shall be permitted as an initial or closing chord in the first or last measure

3rd with tonic being doubled shall be permitted at the beginning or end

3rd with one tone being doubled shall be permitted in the "middle"

Doubling of the leading tone shall not be allowed (leading tone in major & raised 7th in minor - the 7th step of natural minor is not a leading tone)
8.3 Algorithm (pseudo-code)

N.B. “tone” & “note” are used interchangeably.

go() {
    gotOne = false;
    skipOne = false;
    while (!gotOne) {
        while (true) {
            if (!skipOne) {
                allPossibleTonesForNote.add(getNextPossibleTones());
                if (16 notes have been chosen OR the array of possible tones is empty) break;
            }
            Choose a note from the set of currently possible tones.
            Add it to the Line.
            Remove it from the set of currently possible tones.
            skipOne = false;
        }
        if (The line is currently valid)
            gotOne = true;
        else {
            while (The current set of possible notes is empty)
                backTrackOne();
            skipOne = true; (the remaining notes from this array are needed)
        }
    }
}
8.4 Notes:

8.4.1 Redundant CF rules discovered:

- When approaching from below; when previous note is 6th, it is permitted to be raised by half tone
- Step-wise (interval 1 or 2) shall predominate
- Outlining of a 7th within single line moving in same direction shall not be allowed

8.4.2 Redundant CP rules discovered

- Dissonance between ANY 2 parts shall not be allowed; 1,2,10,11+(augmented or diminished intervals)+5(4th)(sometimes)
- A 5th shall be the smallest for 2 voices with one in between
- Upper voices shall be permitted only to be at the 0th, 3rd, 5th, 8th of the tonic
- 1st measure shall contain the tonic in the lowest part
  - Because the CF is the lowest
- Tonic shall be the only note to be doubled
- Any motion shall be used from perfect consonance to an imperfect consonance
- Any motion shall be used from one imperfect consonance to another
• A 3rd shall be the smallest interval between adjacent voices

• Contrary or oblique motion shall be used from one perfect consonance to another

• Contrary or oblique motion shall be used from imperfect consonance to a perfect consonance

• 6th with one tone being doubled shall occur only in the "middle"
  o *Because it cannot occur at the beginning or end*

• If both the 2nd and 7th occur in the upper voices then they shall form the 3rd and 5th of the triad and the lowest shall approach the final tone with a leap of a 4th or 5th
  o *Because CF is necessarily the lowest and the second last is determined; and because the leading tone cannot be doubled*

• <4 successive parallel 3rd,6th,10th
  o *Because >3 parallel “anything” is already handled*