UCLouvain

epl

École polytechnique de Louvain

FuxCP: Constraint Programming Formalisation of Three-Voice Counterpoint According to Fux

Author: **Anton LAMOTTE**
Supervisor: **Peter VAN ROY**
Readers: **Yves DEVILLE, Karim HADDAD, Damien SPROCKEELS**
Academic year 2023–2024
Master [120] in Computer Science and Engineering

Abstract

This master's thesis extends the capabilities of T. Wafflard's FuxCP, a tool designed specifically for composers to assist them in composing two-voice counterpoint without the need for technical expertise. FuxCP uses Gecode and the constraint programming framework to automatically generate counterpoints. The implementation is based on Johann Joseph Fux's *Gradus ad Parnassum*, a seminal treatise on counterpoint published in 1725, by translating the rules of this treatise into formal logic and implementing them as constraints. In particular, the extension to three voices places special emphasis on the lowest voice, introducing innovative concepts and variables to address this key aspect. This thesis contributes to the research and understanding of automated contrapuntal composition by overcoming the challenge of including the lowest voice in the formalisation process and generalising the interaction between voices. Importantly, this work builds seamlessly on T. Wafflard's previous efforts, ensuring full compatibility with his thesis. This research deepens the discussion and experimentation on preferences, a central focus of the thesis. While the concept of preferences was present in the previous thesis, this work delves deeper into their implications. Preferences, treated as optional rules, introduce nuance into the generation of musical solutions, enhancing the overall aesthetic considerations in automated counterpoint composition.

Acknowledgements

…

Contents

Chapter 1

Introduction and context of this work

This thesis is a formalisation for three-part counterpoint based on the writings and rules of Fux. Its aim is to provide a mathematical set of rules and a computer environment capable of translating Fux's teachings into formal logic, and capable of implementing these logical rules in a concrete way to produce Fux-style counterpoint.

This thesis will therefore be divided into several parts: we will first immerse ourselves in *Gradus ad Parnassum*, Fux's central work, from which we will meticulously extract the rules laid down by its author. We will briefly discuss these rules to make them unambiguous, and then translate them into formal logic, so that each rule Fux had in mind when writing his work is mathematically recorded. On this basis, we will create a computer implementation using constraint programming. We will then look at how this implementation finds results, discussing the search algorithm and heuristics used. We then discuss the cost techniques used to obtain the best possible results. Finally, we will analyse the musical compositions produced by the tool created.

It is very important to know that this thesis is based on T. Wafflard's thesis "FuxCP: a constraint programming based tool formalizing Fux's musical theory of counterpoint" [1] and article "A Constraint Formalization of Fux's Counterpoint" [2]. The present work takes up the concepts and definitions of T. Wafflard and could only be understood in its full depth by reading and fully understanding his works as well.

1.1 A brief history of counterpoint: from Bach to algorithmic generation

Before delving into the formalities of our study, let's first examine Fux's theory of counterpoint, which forms the basis of the formalisation undertaken in this work. Counterpoint is a compositional technique in which there are several musical lines (or voices) that are independent and distinct from each other, but that are balanced and sound beautiful [3]. No voice is dominant over the others, and all are main voices, although some may take a small precedence during part of the composition [4].

Counterpoint has been central to the work of many famous composers from different artistic movements, such as Bach in the Baroque era, Mozart in the Classical era and Beethoven in the Romantic era [5]. It is still presenet in some modern music [6], and has aroused interest over the centuries with the development of key texts on the subject, such as Schenker's Counterpoint [7] or Jeppesen's Analysis [8]. And while Bach mastered counterpoint to an unprecedented level for his time [9], the central and foundational work in the teaching of counterpoint belongs to another great Baroque composer: the Austrian Johann Joseph Fux and his treatise *Gradus ad Parnassum*. In it, this composer gives a detailed analysis of the writing of two-, three- and four-part counterpoint, all narrated as a conversation between a master and his pupil. *Gradus ad Parnassum* is one of the works that deal with species counterpoint[1], a way of conceiving counterpoint in five different types that could then be combined. It is on this

[1] There are many other types of counterpoint, such as free counterpoint, dissonant counterpoint, linear counterpoint, ...

work that this dissertation is based.

For Fux, as for many other authors, species counterpoint is governed by many different rules, and it is these rules that interest us in the present work. The rules are based on old concepts that go back to older styles and have been discussed by many other authors [10]. Those concepts include, for example, the notions of opposite motion and consonance (which in turn can be either perfect or imperfect). These concepts and their application to counterpoint are particularly interesting because they allow us to consider the composition of counterpoint both in a 'vertical' way, in which we consider the harmony of the notes played together, and in a 'horizontal' way, in which we consider the melodic development of each of the parts individually, which provides the independence of the counterpoints from each other and their melodic beauty.

This is what makes it interesting to analyse from a constraint programming point of view. We'll come back to this later, but for now let's concentrate on Fux's music theory.

1.1.1 Fux's theory of counterpoint for two-, three- and four-part composition

As we have just said, Fux uses a set of rules to create 'the correct counterpoint'. These rules can be divided into three categories: melodic rules, harmonic rules and motion rules. We will examine them here, bearing in mind that the formalisation of all the rules for two-voice counterpoint can be found in T. Wafflard's thesis, and that the complete set of rules (in mathematical form) can be found in Appendix C of this thesis.

Melodic Rules

Fux explains that there are rules that apply within parts (the horizontal rules) about the order of the interval between one note and the next: we find, for example, that a melody is more "beautiful"[2] when the intervals between its successive notes are small, when there is no chromatic succession, when the successive notes are varied, and so on. These 'horizontal' rules are called 'melodic' rules because they concern only the melody. These rules apply within a given voice.

Harmonic Rules

If there is a horizontal perspective to counterpoint, there is also, of course, a vertical perspective. This perspective is expressed in a harmonic relationship between the different voices. At each point in the composition, a series of rules apply that concern harmony alone (known as harmonic rules). Here are some harmonic rules from Fux, given as an example: the interval between the voices must be a consonance[3] on the first beat of each measure; imperfect consonances[4] are preferred to fifths, which in turn are preferred to octaves; and the voices can't use the same note at the same time. These rules apply between the voices.

[2] Throughout this work we will speak of the "beauty of music". This beauty is highly subjective, and therefore reference will be made to the Fuxian concept of music to define whether a melody is beautiful or not. In other words, music is considered beautiful if it conforms to Fux's rules, and vice versa.

[3] A third, a fifth, a sixth or an octave.

[4] Thirds and sixths.

Motion Rules

Finally, there is a third type of rule: motion rules. These rules are a hybrid of the two discussed above in that they consider not only vertical interaction, i.e. harmony, but also horizontal interaction, i.e. melody. They can therefore be seen as 'diagonal' rules that relate the unique melody of each counterpoint to its respective harmonies. Here are some motion rules, given as an example: voices that move in opposite directions are preferred (i.e. if one voice goes up, we want the other to go down), there should be no successive fifths or successive octaves between voices, and a direct motion should not lead to a perfect consonance. As you can see, these rules take into account not just two voices at a given point, but over several measures. They ensure that the voices move well together.

Preferences

This last point is one of the most important in Fux's theory, namely that of preferences. Preferences are hints that Fux gives in his work in order to write even better counterpoints. As their name suggests, preferences are optional and not compulsory to follow, as other strict rules would be. However, preferences are crucial to Fux's work because they allow us to distinguish between two valid solutions (those that obey all the strict rules) and decide which is the best, thus allowing the composer to control how the theory is applied.

Fux is never clear about whether a rule[5] is a preference or a strict rule — and that's normal, what he conveys is mostly intuition, and human beings are quite capable of understanding whether a rule is a preference or an obligation; Fux probably didn't expect someone to try to formalise his work three centuries later.

These preferences should be respected whenever possible, and if not, so be it. Here's a good example: Fux indicates that we prefer to have as many different notes as possible in the composition. This is not an absolute rule, but a preference. The more variety there is in the composition, the more beautiful it will be, and the more preferable it will be.

1.1.2 Species counterpoint

When we talk about species counterpoint, we're talking about five categories of counterpoint. Each species is a separate concept, each with its own peculiarities. We'll go into more detail about the different species below. Let's focus first on how species counterpoint works. When composing counterpoint, the starting point is a fixed melodic line, the *cantus firmus*, which is a basic melody composed entirely of whole notes. It is the basis of a composition when writing counterpoint. It is from this voice and in relation to it that the others are composed. It is important to note that once the composition is complete, the *cantus firmus* is neither more nor less important than the other voices, and it has the same melodic independence as the other voices. It is, therefore, nothing more than the basis from which we begin to write.

Let's take a look at the five species:

1. **First species**: Note against note – the counterpoint is composed entirely of whole notes, and the composition is a sequence of harmonies sounding on the first beat between the counterpoint and the other voices.

2. **Second species**: Half notes against whole notes – the counterpoint is composed entirely of half notes, which introduce dissonant harmonies.

[5] We use the generic term "rule" to refer to both mandatory rules and preferences.

3. **Third species**: Quarters against whole notes – the counterpoint is made up entirely of quarter notes, which allow more different movements and more freedom in the composition.

4. **Fourth species**: The ligature – the counterpoint is delayed by two beats, creating syncopation. The notes are all round (i.e. tied half notes, since they span between two measures).

5. **Fifth species**: Florid counterpoint – counterpoint is a mixture of all the other species and is the richest form of counterpoint. It allows great freedom of composition while respecting the rules of the other types.

Although these types could technically be combined to form a three-part composition with two different types of counterpoint, Fux seems to prefer to write a 'special' counterpoint (i.e. second, third, fourth or fifth species) in one voice and only whole notes (i.e. a first species counterpoint) in another voice. He also sometimes says that it is possible and recommended to mix species, but does not do so extensively.

1.1.3 Software tool for writing species counterpoint

We have been discussing the longstanding tradition of counterpoint, a musical technique shaped by countless generations of composers. As technology advanced, the idea of automating counterpoint composition emerged. An early attempt, by Schottstaedt in 1984 [11], involved an expert system based on Fux's rules. His approach used over 300 if-else clauses, but this method had obvious limitations compared to what modern constraints are capable of.

Indeed, it's crucial to understand that if-else clauses are unidirectional, whereas constraints are bidirectional. Schottstaedt's algorithm relied heavily on specific conditions, captured by numerous if-else clauses. In contrast, modern constraint systems support bidirectional constraints with multiple arguments. These systems don't just find single solutions, they represent sets of potential solutions. This flexibility is a significant improvement over the directional nature of if-else clauses.

Furthermore, constraint systems offer an advantage in specifying intricate search heuristics. This adaptability and efficiency highlight the stark contrast between the outdated approach of if-else clauses and the modern capabilities of bidirectional constraint systems in the realm of counterpoint composition.

In 1997, a genetic programming and symbiosis approach to automatic counterpoint generation was developed by J. Polito et al. This team from Michigan used the genetic approach to optimise counterpoints of the 5th species and make them more attractive [12]. A similar approach was used in 2004 to generate fugues (hence counterpoint), also using genetic algorithms [13]. The results are quite promising, and generate more than interesting results, but the end result is still far from being able to provide a complete counterpoint composition.

Many years later, in 2010, G. Aguilera et al., from the University of Malaga, developed an automated method for the generation of first-species counterpoint using probabilistic logic [14]. Their approach was specifically tailored to compositions in C major, providing a generated counterpoint in response to a given *cantus firmus*. Please note that this application only evaluates the harmonic attributes of the counterpoint, ignoring the melodic aspect.

Two years later, D. Herremans and K. Sörensen developed a way to generate high-quality first-species counterpoint using a variable neighbourhood search algorithm [15]. Their research was limited to first-species counterpoint, but they addressed issues

such as preferences (finding the best counterpoint) and user-friendly interface. Once again, their results are more than impressive, but their research is limited to the first two-voice species.

Finally, a research was carried out in 2015 on Fux's counterpoint [16], with the aim of generating the first species counterpoint using dominance relations, has yielded fairly good results. The search demonstrates the use of this paradigm and its applicability, and is a good starting point for composing counterpoints of other species based on the same concept.

If we now focus on applications that have gone as far as the user interface and are now ready to use, we should mention two namesakes, both called 'Counterpointer', which have the merit of offering a functional tool for composing counterpoint.

The first Counterpointer tool [17], which anyone[6] can use to check the validity (or not) of their counterpoint. Its last release was in 2019 as a desktop application, and it works like this: an apprentice composer tries to write a counterpoint, and then submits it to the tool. The tool then decides whether the counterpoint is valid according to the traditional rules of counterpoint[7]. It also provides feedback to help the student composer improve their future counterpoint writing. The tool is not able to write counterpoints automatically, nor is it explicit about how it works, as it is completely closed source and has no accessible report. It is therefore impossible to know the paradigm it uses or the exact rules it follows.

Another attempt at automatic counterpoint writing is the Counterpointer project in 2021, created by a team of students at Brown University as part of a software engineering course [18]. The project is less accomplished than the aforementioned application, but it has the merit of being able to generate two-voice counterpoints of the first, second and third species. It is an entirely free and open source project. While the results are encouraging, the project has been discontinued as it was a course project and their method of finding a counterpoint seems much less efficient than the efficiency that a constraint solver can achieve.

This brief overview leads us to conclude that there is no satisfactory tool for composing counterpoint in a user-friendly way, with good quality, quickly and with several voices. It is to fill this gap that this research has been carried out. This was the aim of T. Wafflard's thesis and it is therefore natural that this thesis should follow in his footsteps.

1.2 Standing on the shoulders of giants: underlying works and editions of *Gradus ad Parnassum* used

As has been said, this work is the continuation of T. Wafflard's work. However, it also relies heavily on the work of:

- Lapière [19], who presented an interface for using Gecode functions in Lisp called "GiL". This interface was then tested with some rhythm-oriented constraints.
- Sprockeels [20], who explored the use of constraint programming in OpenMusic using GiL. The tool that was produced in this thesis is capable of producing songs with basic harmonic and melodic constraints.

[6] Anyone... or almost, as it is a paid tool.
[7] Not only Fux's rules, but also those of other authors.

- Chardon, Diels, and Gobbi [21], who created a tool capable of combining the strengths of the first two implementations while continuing to develop support for GiL.

As with T. Wafflard, the musical reference work chosen is Fux's *Gradus ad Parnassum*, because it is a pillar of counterpoint theory and because it is fairly easy to extract rules from it, although Fux is sometimes very vague about his intentions. And as with any book published several centuries ago (1725 in the case of *Gradus ad Parnassum*), there are many versions and translations. This is good news, as Fux can sometimes be really unclear about what he means, and having many versions (some annotated, some not) from many people who also had to interpret Fux to translate it is a great treasure, as it helps to clarify Fux's meanings. This work is therefore based on several different editions and translations of the book, although it is mainly based on Alfred Mann's English translation [22]. French (both Chevalier's [23] and Denis's [24]), German [25] and Latin [26] translations are used when it is necessary to remove an ambiguity or clarify an unclear rule. These translations have been chosen because French is the lingua franca of the team; German is the language of Fux and the environment in which he evolved; and Latin is the original version, so we can hope that it is the most faithful to what he wanted to convey.

1.3 Tools and implementation

This subsection discusses the implementation of the automatic counterpoint generator. To do so, we briefly explain how constraint programming works, the tools used by FuxCP, and the actual implementation of it.

1.3.1 Constraint Programming

Constraint Programming (CP) is a programming paradigm used to solve large combinatorial problems, such as planning and scheduling problems. It works by defining constraints between variables that limit the values these variables can potentially take [27]. In doing so, the domains of these variables are reduced. R. Barták explains in a very clear way what a constraint really is, as he describes in his "Guide to Constraint Programming" [28]:

> A constraint is simply a logical relation among several unknowns (or variables), each taking a value in a given domain. A constraint thus restricts the possible values that variables can take, it represents some partial information about the variables of interest. For instance, "the circle is inside the square" relates two objects without precisely specifying their positions, i.e., their coordinates. Now, one may move the square or the circle and he or she is still able to maintain the relation between these two objects. Also, one may want to add other object, say triangle, and introduce another constraint, say "square is to the left of the triangle". From the user (human) point of view, everything remains absolutely transparent.

The question now is: what is the connection between this problem-solving method and counterpoint composition? Well, it turns out that music is actually a very good application of constraint programming: you can represent all aspects of a composition by variables, set rules between these variables, and the solver takes care of finding a valid solution. In fact, in music, it's never just one factor that determines whether the music is beautiful, but an interaction of many. And as humans, it's sometimes difficult

to find a valid solution (i.e. to compose music that sounds good) because the range of possibilities and the interactions between factors are so numerous. However exploring a search space in which a large number of constraints are defined is something that a constraint solver does very well. The most arduous task then becomes putting the rules that make music beautiful down on paper, and this is the task that many musicologists and composers, like Fux, have set themselves. Once these rules have been defined, it is "simply" a matter of formalising them and passing them to the constraint solver so that it can compose a melody that respects these rules.

What's more, by defining a rigorous way of distinguishing a good composition from a bad one, the solver can even find increasingly beautiful solutions.

To make sure that Constraint Programming is a well understood concept, we here review its main concepts:

Constraint propagation Each time a constraint is defined, the domain of the affected variables is reduced according to the possibilities left by the constraint. This is called constraint propagation. Let's imagine a variable A whose domain is $\{0, 1, 2, 3\}$ and a variable B whose domain is $\{0, 1, 2\}$. When the constraint $A < B$ is applied, the domain of A is reduced to $\{0, 1\}$, because the new constraint makes it impossible for A to have a value of 2 or 3 without violating the constraint.

A constraint solver is an implementation that systematically browses the search space in look for a solution. A given problem can have many solutions, just one, or even none, if the domains are too small or the constraints contradict each other, and the system becomes overconstrained. A solution is found when all variables are fixed, i.e. their domain is reduced to a single value. We know that a problem has no solution if a domain is empty after a constraint has been declared (because this means that no value can be found for that variable that does not violate the constraint).

Branching Obviously, declaring constraints is not enough to magically find a solution. In the example we proposed earlier (with A and B), the single constraint placed on the search space doesn't allow us to determine the values of A or B, and their domains remain composed of more than one value. To actually find a solution, the constraint performs a branching. That is, it studies two antagonic possibilities and splits the search space into two subproblems accordingly. More specifically, it chooses a variable and studies the case where that variable is equal to a certain value, and the case where it is not equal to that value. For example, the solver might study the case where $B = 0$ (the first branch) and the case where $B \neq 0$ (the second branch). If the solver finds an inconsistency in either case, it knows that the entire branch can be discarded (as it does not lead to a valid solution). Immediately after branching, the solver again performs constraint propagation, since constraint propagation occurs whenever any domain is modified, and consists of adjusting all domains to which the modified domain is linked by a constraint. In our case, after setting B to 0, the solver propagates all constraints linked to B, i.e. the only constraint in our problem (i.e. $A < B$). This affects the domain of A, reducing it to an empty domain (because no value in the domain of A is less than 0). The solver, noticing that one domain is empty, concludes that this branch contains no solution and therefore knows that the only possible branch is the other one, i.e. the one in which we assumed $B \neq 0$. We can therefore safely remove 0 from the domain of B, since this value is not contained in any solution. Repeating the branching process each time it is necessary produces three solutions: $A = 0$ and $B = 1$, $A = 0$ and $B = 2$, and finally $A = 1$ and $B = 2$.

Heuristics As with all problems involving searching a space in quest of a solution, it is very useful to have heuristics allowing for an efficient search. A heuristic is a rule or strategy used to make informed decisions about variable assignments and value choices during the solution search. These rules are designed to exploit the characteristics and structures of the problem to improve the chances of finding solutions more quickly. A common heuristic is variable ordering, where the algorithm selects variables to assign values to based on factors such as the size of the domain or the number of associated constraints. Another important heuristic is value ordering, which determines the order in which values are tested for a given variable assignment. By incorporating heuristics, constraint solvers can prioritise the most promising branches of the search tree, effectively reducing the search space and speeding up the identification of feasible solutions. While heuristics speed up the solving process, it's important to strike a balance between exploration and exploitation, as overly aggressive heuristics risk missing potentially valuable solution paths.

To return to our previous example, a value-ordering heuristic might be "branch first on low values of A and high values of B, since we know that we are looking for a solution where A is less than B, and we can reason that there are more chances of satisfying this constraint when B is large and A is small.

Costs in Constraint Programming While the basis of Constraint Programming is this succession of propagations and branchings, there also exists a notion of cost, which distinguishes between two valid solutions (i.e. satisfying all constraints) into a less good solution and a better solution.

For example, in our simple example, we could define cost as the sum of A and B and want to minimise the cost. This means that we are looking for a valid solution where A and B are as small as possible. In this case, only the first of the three solutions mentioned above is chosen, i.e. $A = 0$ and $B = 1$, as it is the best possible solution (with a cost of 1).

Branch and Bound Branch and Bound is a systematic algorithm used in Constraint Programming to efficiently explore the solution space and find an optimal solution with respect to a given cost or objective function. This technique extends the basic Constraint Programming approach by introducing a mechanism to prune unpromising branches of the search tree, thereby reducing the computational effort required to find the optimal solution. The algorithm starts with the initial problem and iteratively divides it into subproblems, called branches, by making decisions about variable assignments. For each branch, the algorithm evaluates its feasibility and its potential to lead to a better solution. If a branch is deemed infeasible or cannot possibly improve on the current best-known solution, it is pruned from further consideration. This process continues until all branches have been explored, or until the algorithm converges on the optimal solution. Branch and Bound is particularly valuable for large combinatorial problems because it efficiently narrows the search space, allowing the solver to focus on promising regions and accelerating the discovery of optimal solutions in constrained programming scenarios [29].

1.3.2 OpenMusic

OpenMusic is a powerful and innovative visual programming environment, written in CommonLisp, designed specifically for composers, researchers and musicians involved in computer-aided composition [30]. Developed by the Institute for Research and Coordination in Acoustics/Music (IRCAM) in Paris, OpenMusic provides a graph-

ical interface that allows users to create and manipulate musical structures using a variety of predefined modules. This visual programming language facilitates the representation of complex musical ideas, algorithms and data flows through a user-friendly interface, making it accessible to both novice and experienced composers.

FuxCP is a library for OpenMusic, which means that OpenMusic is the interface for using FuxCP. Any user wishing to use FuxCP writes their *cantus firmus* (FuxCP's input) into OpenMusic and then launches the solution search from within OpenMusic. Specifically, FuxCP retrieves the *cantus firmus* from OpenMusic and then defines the constraint problem. When a solution is found, it is passed to OpenMusic and the user gets it as an OpenMusic object.

1.3.3 Gecode and GiL

After receiving the input (which is the *cantus firmus*), FuxCP uses Gecode to define the search space and starts searching for a solution. Gecode, short for Generic Constraint Development Environment, is an open source toolkit for developing constraint-based systems [31]. It provides a high-level C++ library for efficiently modelling and solving constraint problems. Gecode supports a wide range of constraints, variables, and search strategies, making it a versatile platform for tackling combinatorial problems such as scheduling, optimisation, and configuration.

Since FuxCP is an OpenMusic library written in Lisp, it is naturally written in Lisp too. To be able to use the Gecode C++ library, FuxCP uses the Gecode Interface Lisp (GiL), which is a wrapper that allows Gecode functions to be used in Lisp programs. It in turn relies on The Common Foreign Function Interface (CFFI), which is an interface for calling C++ functions in Lisp [32]. GiL is far from complete, but it provides enough tools to solve interesting constraint problems in Common Lisp.

1.3.4 Concrete implementation

To put it all together: FuxCP gets its input (the *cantus firmus*) from the user interface, which is OpenMusic. It then defines a constraint programming problem in Gecode, using GiL.

As for the way it defines the problem, here is a little clarification: first, when the *cantus firmus* is received, a whole series of constants and variables are defined: for example, the length of the *cantus firmus*, the arrays representing the pitches of the counterpoints, ... Then all these variables are constrained according to the constraints defined in the formalisation of Fux's rules (discussed in chapter 3). The constraints are set sequentially: first the constraints on the *cantus firmus*, then the constraints on the first counterpoint, and finally the constraints on the second counterpoint. A diagram of the code architecture and the integration of FuxCP with other tools can be found in Appendix A.2.

1.4 The contributions of this thesis

The aim of this work is to generalise T. Wafflard's formalisation to three-voice counterpoint, still based on Fux's work, and to create the corresponding implementation. It would be too easy to believe (wrongly) that three-voice counterpoint is nothing more than the combination of two two-voice counterpoints. From this point of view, we would then calculate a first counterpoint according to the *cantus firmus*, and then a second counterpoint again according to the *cantus firmus*, and that's it. Obviously, this view is too simplistic and doesn't really capture all the interactions between three voices. It is to this point (the peculiarities brought about by the addition of a third

voice) that a whole chapter of this thesis is devoted. Another chapter is dedicated to translating the rules from *Gradus ad Parnassum* into formal logic. A final but not less important section discusses and analysing the impact of costs, and the musicality of the solutions. The following is a more detailed summary of the contributions of this thesis.

- **Concepts and variables to three-part counterpoint**: As we have just mentioned, a three-part composition is much more than a (two+one) part composition. So we had to redefine and define a whole series of concepts to adapt to this reality. The creation of the (lowest, middle and highest) stratum concept is part of this, and is essential for formalising Fux's counterpoint constraints. All of this is discussed in Chapter 2.

- **Mathematical formalisation of three-part counterpoint**: As with the two parts of the formalisation, we rewrote Fux's explanations into unambiguous English and then translated them into logical notation. This formalisation builds on the previous formalisation for two voices, and sometimes (rarely) has to modify it. This formalisation can be found in Chapter 3.

- **Implementation of a working constraint solver for a three-voice composition**: Those logical rules were then implemented as constraints and the solver was adapted to allow a search for two counterpoints. The whole code of this implementation can be found in Appendix D, and its architecture in the Appendix A.

- **Researching the best way to express Fux's preferences**: Three-part composition introduces so many possibilities for result composition that it is important to rethink the way we think about preferences. These preferences are understood by the solvers as costs (where a preferred solution in Fux's sense has a lower cost to the solver). Therefore, some techniques for managing these preferences are discussed to find out the best way to implement them as costs. This is very important as it allows the solver to produce solutions with high musicality. These techniques are discussed in Chapter 4.

- **Musical analysis of the solutions generated by the solver**: Finding the best solution also means being able to assess the quality of current solutions. For this, see Chapter 5.

- **User interface for three-point counterpoint that allows a composer to specify how preferences are used in the solver.**: All the new capabilities of the solver and the costing techniques must also be accessible to the user: it is now possible for a user to freely combine any number of species to form a three-part composition, and to set a cost importance order to indicate their preferences to the solver (in addition to the already existing ability to set personalised costs). A guide to its installation and use can be found in Appendix B.

Chapter 2

Defining some concepts and redefining the variables

2.1 Voices, parts and strata

Before we start this section, we need to look at some vocabulary to make sure we understand what we are discussing. The most important definitions we introduce are the concepts of *parts* and *strata*. The need for these definitions arises from the increasing complexity of the rules of counterpoint when it is generalised to three voices. Indeed, the rules are no longer (as we shall see later) concerned solely with the counterpoints and the *cantus firmus*, but also with new concepts, such as that referred to by Fux as 'the lowest voice'. As the term 'voice' is too generic (it is used in Fux's text to describe notions as different as 'counterpoint', '*cantus firmus*', voice range and the so called 'lowest voice'), we need to create a precise vocabulary that is different from the word 'voice' to talk about these new concepts.

With this in mind, let's explain what 'parts' and 'strata' are, and how they relate to the concept of 'voice'.

Voices

Again, voices are that vague and *general* concept, whereas parts and strata are more precise and *specific* concepts. The concept of 'voice' includes both 'parts' and 'strata'. In other words, each of these two concepts is a type of voice. When we talk about a voice, we could be talking about either a part or a stratum. To use an object-oriented metaphor, we could say that the 'parts' and 'strata' classes inherit from the 'voice' class.

Since there are as many parts and layers as there are voices, in a composition with n voices there will also be n parts and n layers.

Parts

Parts are an intuitive and concrete concept because each part corresponds to what a particular person sings or what a particular instrument plays. They correspond to a staff (each staff corresponds to a part). The term 'part' is the same as that used by Fux in his *Gradus ad Parnassum*. The three parts in a three-part composition are: the *cantus firmus*, the first counterpoint and the second counterpoint. Fux distinguishes them by calling them by the name of their range, i.e. "bass", "tenor", "alto" or "soprano" (obviously you cannot have all four in a three-part composition).

Strata

As for the strata, they are defined like this: a stratum delineates discrete layers or levels of pitches at any given moment in the composition. It denotes a vertical alignment of simultaneous notes and organizes them into distinct strata. By definition, the lowest stratum encompasses the lowest sounding notes, the highest stratum comprises the highest sounding notes, and intermediary strata represent pitch levels in between.

This concept is very helpful in identifying and categorising the vertical placement of pitches, creating distinct categories of sound within the overall texture of the counterpoint composition. It provides a way of analysing and understanding the distribution of pitches across different parts, allowing more complex rules to be established. For example, it would now be possible to establish a rule between the notes of the cantus firmus and the highest sounding notes (no matter which part they come from). The full potential of strata lies in harmonic rules, but as we shall see, some melodic rules are also related to it.

Important note concerning the strata Strata are an abstract concept, useful only in the mathematical formalisation of Fux's rules. They are necessary because we need a structure that is able to comprehend the lowest sounding note for each bar. The strata concept is obviously not needed to write counterpoint as a human being, and the aim behind its definition is not to create a new concept for music theory, but to enable us to use a tool in our constraint programming way of conceiving counterpoint composition.

The term stratum was chosen in this context for its visual impact. In geology, a 'stratum' "is a rock layer with a lithology (texture, color, grain size, composition, fossils, etc.) different from the adjacent ones" [33], see figure 2.1.

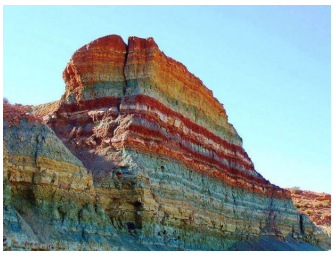

Figure 2.1: Geological strata, for illustration

When Fux speaks about the lowest stratum, he often uses the word 'bass'. It was deliberately chosen to speak about the 'lowest stratum' instead of the 'bass' (like Fux does), because 'bass' is also the name of a range of voices (like soprano and alto, for example), and there is already enough complexity in all the terminology to add even further ambiguity.

These new terms (parts and strata) are used where the distinction between the concepts is important. Whenever this distinction is not relevant, the more general term 'voice' is used to reduce the complexity of reading. In this case, the 'voice' could refer to either a stratum or a part. And since a picture is worth a thousand words, Figure 2.2 illustrates the difference between parts (the blue lines) and strata (the red and orange lines). The lowest stratum is shown in its own colour (red) because it is the most meaningful stratum, and it is particularly important in the formalisation.

Here is also the mathematical representation for the notes of the lowest stratum (written $N(a)$, see section 2.3 for the notations):

$$\forall i \in [0,3] \quad \forall j \in [0, m-1) : N(a)[i,j] = \min(N(cf)[i,j], N(cp_1)[i,j], N(cp_2)[i,j]) \tag{2.1}$$

The first upper stratum, or medium stratum (written $N(b)$, see section 2.3 for the notations):

$$\forall i \in [0,3] \quad \forall j \in [0, m-1) : N(b)[i,j] = \text{med}^1(N(cf)[i,j], N(cp_1)[i,j], N(cp_2)[i,j]) \tag{2.2}$$

[1]Where med(X) means the median value of X.

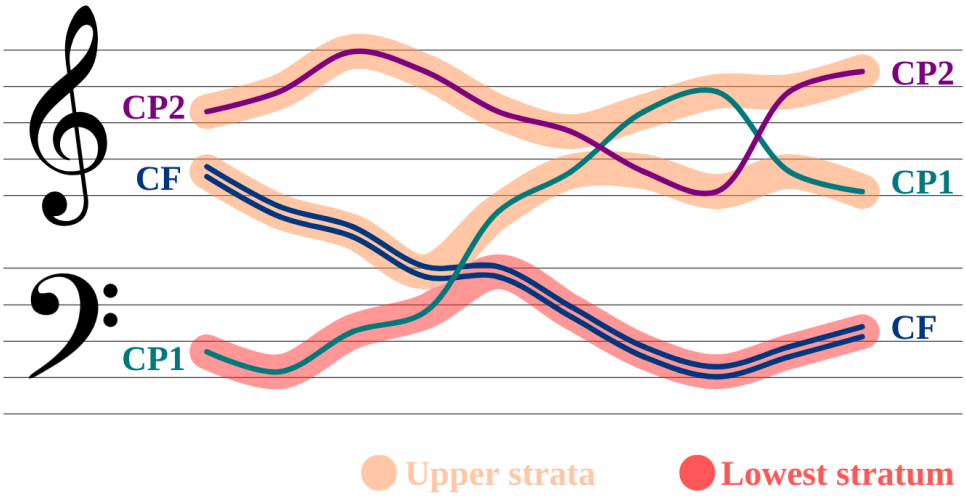

Figure 2.2: Parts and strata in a three voice composition

The second upper stratum, or uppermost stratum (written $N(c)$, see section 2.3 for the notations):

$$\forall i \in [0,3] \quad \forall j \in [0, m-1) : N(c)[i,j] = \max(N(cf)[i,j], N(cp_1)[i,j], N(cp_2)[i,j]) \tag{2.3}$$

One part per stratum and one stratum per part

It is important to note that, for each musical measure, there is a bijection between the individual parts and the corresponding strata. This means that, for any given measure, each stratum uniquely corresponds to a single part, and vice versa. Put differently, if two parts within a measure share the same pitch, they do not constitute the same stratum. Instead, one part corresponds with one stratum, and the other one to a separate stratum.

To illustrate this, consider a scenario in a two-voice composition (see figure 2.3), where part 'cf' and part 'cp1' in measure X both have a pitch value of 67 (representing a G). Despite having identical pitches at the same moment, one part is categorised as the lowest stratum, while the other is designated as the uppermost stratum. This distinction becomes crucial for subsequent analysis, especially when calculating aspects like motions.

To know which part gets to be the lowest stratum in such situations, an arbitrary hierarchical rule is implemented. If the ambivalence is between the *cantus firmus* and another part, the *cantus firmus* is always prioritised and assigned the role of the lowest stratum, over any other part. In the case of a ambivalence between the first counterpoint and the second counterpoint, the first counterpoint is given the status of the lowest stratum.

Note concerning the intersection of the voices

This new stratum concept makes it possible to create compositions in which the bass doesn't always play the lowest notes, and this is precisely its purpose. Note, however,

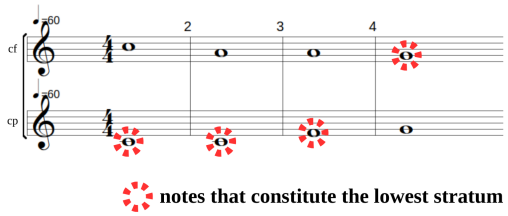

Figure 2.3: Establishing which part corresponds to the lowest stratum

that in order for a voice other than the bass (the tenor, for example) to play the lowest note, the melodies of the tenor and the bass must cross. Fux is strangely silent on this subject: he says that crossings are perfectly permissible, but he doesn't elaborate. Other authors are clearer, and the rules of species counterpoint generally state that crossing is allowed, but that the voices must not remain inverted for too long [34, p.28]. However, this is a rule that was not taken into account in the formalisation because it is not present in *Gradus ad Parnassum*. It is therefore a way of improving the FuxCP tool, with the aim of making it compatible with other styles of counterpoint.

2.2 Exploring the interaction of the parts with the lowest stratum

One of the major differences between the composition of two voices (i.e., one *cantus firmus* and one counterpoint) and the generalisation to three voices (i.e., one *cantus firmus* and two counterpoints) is that the rules no longer necessarily apply between the counterpoints and the *cantus firmus*, but instead of this are mostly applied **between the different parts and the lowest stratum**.

If we go back to the rules for two voices, we see that each of them applied between the single counterpoint and the *cantus firmus*. For example, when it was stated that each interval must be consonant, this referred to the harmonic interval between the counterpoint and the *cantus firmus*. On the other hand, in his second part (where he describes the rules for composing in three voices), Fux explains that the rules do not necessarily have to be followed between each counterpoint and the *cantus firmus*, but rather between "each of the voices and the lowest voice" (i.e. the lowest stratum). Again, if we take the example of the need for consonance between the voices, consonance will be required in the intervals between the notes of any voice and those of the lowest voice (whether or not the latter is the *cantus firmus*). Fux approaches the concept of lowest stratum without ever stating it clearly, mentioning for example that the lowest voice can change (sometimes the bass is the lowest voice, sometimes the tenor, ...), and that at any given moment the lowest voice should be considered. In other words, Fux says that the rules apply between the parts and the lowest stratum.

In summary, the constraints are as follows:

- Most of the constraints apply:
  - Between the *cantus firmus* and the lowest stratum.
  - Between the first counterpoint and the lowest stratum.
  - Between the second counterpoint and the lowest stratum.

- Some constraints apply:
  - Between the *cantus firmus* and the first counterpoint.
  - Between the *cantus firmus* and the second counterpoint.
  - Between the first counterpoint and the second counterpoint.
  - Between the three parts altogether (harmonic rules only).

Generalisation of two-voice counterpoint

One might be tempted to conclude that three-part composition breaks completely with two-part composition, but that would be too hasty a conclusion. Indeed, on closer inspection, the way the rules worked in two-part composition (from counterpoint to *cantus firmus*) is just one particular case of this new vision of things. In two-part composition, too, the rules apply between the parts and the lowest stratum. But of course, since there were only two voices, the lowest stratum was either counterpoint or cantus firmus. This means that when links were established between the upper part and the lowest stratum, links were also established between the counterpoint and the cantus firmus. Considering the rules as being established between the counterpoint and the *cantus firmus* was just a simplification of reality, although it was perfectly correct. We were therefore considering a convenient particular case, and not the general case. Please note that when we talk about "applying constraints from voice A to voice B", it is clear that the constraints are bidirectional and that they also apply from voice B to voice A. What is shown here is rather the philosophy behind the application of these constraints, and the reasons why they were imposed.

The particular case happening when composing with two parts is illustrated in figures 2.4 and 2.5. As we can see on those pseudo-compositions, it does not change anything to apply the constraints between the counterpoints and the *cantus firmus* or between the parts and the lowest stratum.

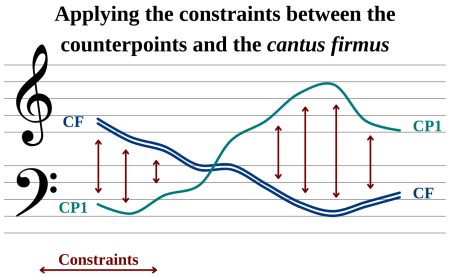

Figure 2.4: Applying the constraints between the counterpoint and the *cantus firmus*

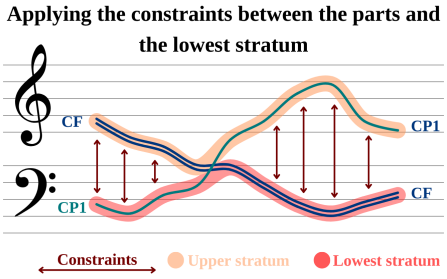

Figure 2.5: Applying the constraints between the parts and the lowest stratum

However, when it comes to generalising the composition of counterpoint for three voices, the same simplification is no longer possible. We are now forced to establish our rules between the parts and the lowest stratum, and no longer between the counterpoints and the *cantus firmus*. In figures 2.6 and 2.7 it becomes clear that establishing the rules between the counterpoints and the *cantus firmus* is really different from applying them between the various parts to the lowest stratum. In these figures, the parts don't intersect and therefore fit perfectly with the strata, so the constraints are always applied to the same counterpoint. This was done for the sake of intelligibility of the graphs, but it is of course possible for the parts to cross and for the "target" of the constraints not always to be the same counterpoint.

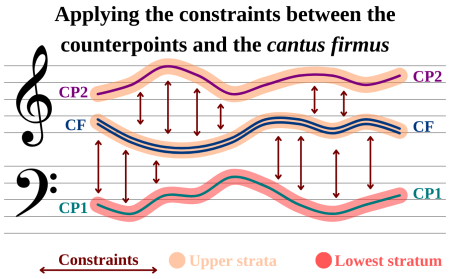

Figure 2.6: Wrong approach: applying the constraints between the counterpoint to the *cantus firmus*.

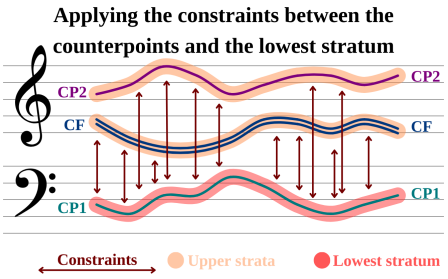

Figure 2.7: Correct approach: applying the constraints between the parts to the lowest stratum.

It is, of course, possible for the *cantus firmus* to be equal to the lowest stratum all along, in which case nothing changes from the perspective we had when composing for two voices. In this particular case, by applying the rules with respect to the *cantus firmus*, we would find ourselves de facto applying the rules with respect to the lowest stratum (and we would be back to the situation described above, see figures 2.6 and 2.7, only that there is now one more part). It is when the *cantus firmus* pitches are higher up than those of the counterpoints that considering the lowest stratum consideration becomes necessary.

A very important detail, and perhaps the biggest change brought about by this paradigm shift, is the following. Previously, we applied constraints between the counterpoints and the *cantus firmus*, which guaranteed that the *cantus firmus* was taken into account in the constraints. But if we now apply the constraints only between the counterpoints and the lowest stratum, there is no longer any guarantee that the *cantus firmus* will be linked to the other voices by any constraints, for example if the *cantus firmus* is not the lowest stratum. Nevertheless, it is important that the relationship between the *cantus firmus* and the lowest stratum is *also* taken into account, not just the relationship between the counterpoints and the lowest stratum. This means that when we apply the constraints between the parts and the lowest stratum, we *must* also apply them to the *cantus firmus* (since the *cantus firmus* is a part, like any of the counterpoints), unless explicitly stated otherwise.

A second point to bear in mind, and not the least, is that all this does not mean that *all* the rules are established between the parts and the lowest stratum. Certain rules continue to apply between the different parts, regardless of whether they are high, low or intermediate.

2.3 Definitions of the variables used in the formalisation

In this section we define the variables used in the formalisation. Many of these variables were already present in T. Wafflard's work and are reused in this formalisation, with some changes. All these (re)definitions are explained in detail in this section.

Throughout this section, when reference is made to the past ("this variable used to be", "this variable keeps the same definition", ...), it means that reference is made to the previous definition of the variable, which was the one defined in T. Wafflard's work.

Nota bene Please take into consideration that all the rules from T. Wafflard's thesis (which can be found in Appendix C) are fully compatible with the new definitions of

the variables, as discussed in 2.2.

2.3.1 Variables and array notation

First, let's look at how the variables are defined and the notation we use to access them. The variables are defined to capture a compositional reality, such as the notes of the voices, the harmonic intervals between those voices, or many other concepts. They are represented by a letter (N for the notes, H for the harmonic intervals, ...) and are mainly arrays because they have a different value for each beat of the composition. To know which beat we are talking about, we address the variable in a computer notation. For example, N[i, j] means "the note on the i-th beat of the j-th measure". This implies that i ranges from 0 to 3 and that j ranges from 0 to the number of measures (which is written as m).

Once defined, the variables are related to each other according to the formalised rules. The constraint solver searches for all possible values of these variables, according to the constraints, and stops when all variables in N (the pitches) are fixed, as this means that a solution has been reached (the notes of the counterpoints are known, and this is the goal of the solver).

Of course, there are many different solutions for the same *cantus firmus*, and in order to distinguish between two valid solutions (i.e. all solutions that respect all constraints). So some variables are actually costs, that are intended to convey the preferences expressed by Fux in *Gradus ad Parnassum*. Thus, the solver considers a valid solution with a low cost to be better than a valid solution with a high cost. The costs can be 0 (no cost), 1 (low cost), 2 (medium cost), 4 (high cost), 8 (last resort) and $64m$ (cost proportional to length). These costs are then combined into a total cost that the solver tries to minimise. We will see in chapter 4 how the costs are actually combined.

2.3.2 Overview of all the variables

We may now take a look at all the variables that are useful throughout this work. You may note that a vast majority of them already exists in T. Wafflard's thesis, but with one big change, each variable is now linked to a voice. To understand this, let's take an example. In a two-part composition, it was obvious that H (the harmonic intervals array) described the intervals between the *cantus firmus* and the only counterpoint. It was also obvious that P (the motions array) described the motions of the single counterpoint. And so it is with all the variables. When writing a three-voice composition, we have many possibilities when we talk about intervals or motions. Intervals between which voices? Movements of which counterpoint? To deal with this, each variable is now related to a voice.

The relationship between a variable and a voice is expressed as a function. $X(v)$ represents the variable X of the voice v. The arguments of the function can be either:

- cf - for linking the variable to the *cantus firmus*.
- cp_1 - for linking the variable to the second counterpoint.
- cp_2 - for linking the variable to the third counterpoint.
- a - for linking the variable to the lowest stratum.
- b - for linking the variable to the intermediate stratum.
- c - for linking the variable to the uppermost stratum.

For example, $X(cf)$ refers to the variable X of the *cantus firmus*.

When a variable is not explicitly linked to a voice, it is implied that the relation expressed for it is true for all *parts*. In other words, if the variable X is written without any precision, it means that we are speaking about the variable X of all parts. Formally, $X \equiv \forall v \in \{cf, cp_1, cp_2\} : X(v)$. This is very important, as it makes it possible for T. Wafflard's rules for two voices to *remain valid* even with the new definition of the variables.

As mentioned before, linking the variables and the voices is something that applies to all variables, namely[2]:

- **N**(v) - the notes (pitches) of the voice v. This is the same variable as the variable 'cp' in T. Wafflard's thesis (an explanation of the renaming can be found in the corresponding section (section **??**)).
- **H**(v_1, v_2) - the harmonic intervals between voice v_1 and voice v_2. This variable is particular, as it needs two arguments to be meaningful.
- **M**(v) - the melodic intervals of the voice v,
- **P**(v) - the motions of the voice v,
- **A**(p) - the boolean array representing whether the part p is the lowest stratum.
- **IsCfB**(v) - the boolean array representing whether the cantus firmus is lower than the voice v,
- **IsCons**(v) - to the boolean array representing whether the voice v is consonant with the lowest stratum or not,
- **S**(p) - an array specific to the fifth species, representing for each beat what set of rules a note follows. For example $S[0, 0] = 3$ means that the very first note of the fifth species counterpoint actually belongs to the third species. This is important since the fifth species is actually a mixture of all the others.

It also applies to *some* constants, namely:

- **species**(p) - the species of part p — by definition, $species(cf) = 0$,
- **n**(p) - the number of notes in part p,
- $\mathbf{s_m}$(p) - the maximum number of notes contained in part p, if all notes were quarter notes, excepted the last one (that is always a whole note),
- **lb**(p) - the lower bound of the range of part p,
- **ub**(p) - the upper bound of the range of part p,
- $\mathcal{R}$(p) - the range of part p, i.e. $\mathcal{R}(p) := [lb(p), ub(p)]$,
- **borrow**(p) - the borrowing scale[3] of part p,

[2]This list contains all the variables used in this thesis and a short description of them. If no formal definition or redefinition is mentioned in this work, it means that the applicable definition is the one given in T. Wafflard's thesis.

[3]Remember that Fux mainly uses notes without a flat or sharp, which means that if the composition begins with a C, it will be written in Ionian mode, if it begins with a G, in Mixolydian mode, etc. Borrowing mode is a way for counterpoint to access notes that are not originally in its mode. For example, a counterpoint in E (i.e. Phrygian mode) that has the major mode (E major, i.e. Ionian E) as its borrowing mode will also have access to the notes F#, C#, G# and D#.

- $\mathcal{N}$(p) - the borrowed notes of part p, where $\mathcal{N}^{\mathcal{R}}(p) = \mathcal{N} \cap \mathcal{R}$
- $\mathcal{B}$(p) - the set of beats[4] in a measure according to the species of part p,
- b(p) - the number of beats[5] in a measure according to the species of part p,
- d(p) - the duration of a note[6] according to the species of part p,
- m - the number of measures in the composition, being a whole note.

Please note that the constants can only be linked to the parts, never to a stratum. Indeed, it would have no sense to speak about the species of a stratum or about the extended domain of a stratum.

The costs are also affected by the linking, except for $\mathcal{C}$ (the cost factors) and τ (the total cost). The latter are global variables that exist only once. For a summary of all costs, please refer to Table B.1.

To make sure that those notations are clear, here are some examples: the notation $N(a)$ corresponds to the variable representing the notes (pitches) of the lowest stratum, whereas $N(cf)$ are the notes of the *cantus firmus*. The species of the second counterpoint is written $species(cp_2)$. If only N is written, then the equation in which N is located holds true for any possible *part*. That is, the relationship $N[0,0] < 60$ would mean: the pitch of the first note *of all parts* must be lower than a middle C (whose representation in MIDI is 60).

Note regarding the fourth species

Let's recall that the fourth species behaves in a particular way compared to the other species. First of all, it is exclusively composed of syncopations. Its notes are half notes, always linked two by two from bar to bar, producing a pitch change in the middle of the measure, on the upbeat. This gives the impression of hearing a whole note that is constantly shifted by two beats, in other words: syncopation.

Concretely, and as Fux explains it, the syncopation means that the beats of the fourth species should be considered as "shifted": its upbeat should be considered as the downbeat, and its downbeat as the upbeat of the previous measure. This means that in the majority of cases, the equations for the fourth species would have to be rewritten, swapping the 0 and 2 indexes (H[2, j] becomes H[0, j] and H[0, j+1] becomes H[2, j]). To avoid duplicating each of the equations (a first equation if it is not of the fourth species and a second equation if it is of the fourth species) and also to avoid equations that are too complex and difficult to read, it was decided that the index swap would be implicit.

Here is an example: $H[0,0] = 7$ should be understood as $H[2,0] = 7$ if it concerns a fourth-species counterpoint.

[4]To make it clearer: for the first species, the only beat in a measure is $\{0\}$, as there is only a note on the first beat. For the second species, the set of beats is $\{0, 2\}$. For the third species, it is: $\{0, 1, 2, 3\}$. For the fourth species: $\{0, 2\}$. And for the fifth species: $\{0, 1, 2, 3\}$.

[5]Thus, it is always equal to the size of the set $\mathcal{B}$(p).

[6]For the first species, it is equal to 1, as each note is a whole note. For the second species, it is $\frac{1}{2}$, for the third, it is $\frac{1}{4}$, for the fourth, it is $\frac{1}{2}$, and for the fifth, it is $\frac{1}{4}$. It is always equal to $\frac{1}{b(p)}$.

2.3.3 In depth definition of the variables

N(v) `notes`

N is the array corresponding the pitches of each voice. Its size is s_m. It is the same array as the one named `cp` in T. Wafflard's thesis, and it got renamed to `N` (for notes), for the sake of clarity. As we have now three of those arrays (one for the first counterpoint, one for the second counterpoint, and even one for the *cantus firmus*), it needed a less ambiguous name than the one it had before.

H$_{\text{(abs)}}$(v$_1$, v$_2$) `h-intervals h-intervals-abs h-intervals-to-cf` ...

This variable is an array of size s_m and represents the harmonic intervals between voice v$_1$ and voice v$_2$. It is the only variable that is associated with two different voices. So $H(v1, v2)[i, j]$ represents the intervals between the ith beat of voice v_1 and the *first* beat of voice v_2. v_1 can be a part and v_2 can be a stratum, since you can calculate harmonic intervals between a part and a stratum. If v_2 is not specified, it is equal to a by default. In other words, $H(v_1)$ represents the intervals between the voice v_1 and the lowest stratum: $H(v_1) \equiv H(v_1, a)$. This default value for v_2 was chosen because it is the most commonly used, and for a good reason: the most relevant harmonic intervals are those between the voices and the lowest layer.

$$\begin{aligned}
&\forall v_1, v_2 \in \{cf, cp_1, cp_2, a, b, c\}, \quad \forall i \in \mathcal{B}(v_1), \quad \forall j \in [0, m) : \\
&H_{abs}(v_1, v_2)[i, j] = |N(v_1)[i, j] - N(v_2)[0, j]| \\
&H(v_1 - v_2)[i, j] = H_{abs}[i, j] \bmod 12 \\
&\text{where } H_{abs}[i, j] \in [0, 127], H[i, j] \in [0, 11]
\end{aligned} \tag{2.4}$$

M$_{\text{brut}}$(v) `m-intervals-brut`

The variable M represents the melodic intervals of a voice. It can be evaluated either on a part or on a stratum, each of these situations leading to different behaviours.

- M(p) (i.e. when related to a part) represents the melodic intervals of the part p, and its mode of operation remains the same as in T. Wafflard's thesis: M$_{brut}$[i,j] represents the melodic interval used to "leave" N[i,j]. For example, if $N[0, 0] = 60$ and $N[1, 0] = 67$, the interval used to leave $N[0, 0]$ is 7. There are several versions of this array, such as M^x, which represents the melodic interval between the current beat and the xth following note, according to d(p)[7] and M, which represents the absolute value of M$_{brut}$. By default, $M \equiv M^1$, thus, $M \equiv \forall p \in \{cf, cp_1, cp_2\} : M^1(p)$.

$$\begin{aligned}
&\forall x \in \{1, 2\}, \forall i \in \mathcal{B}, \forall j \in [0, m - x) : \\
&M^x_{brut}[i, j] = N[(i + xd) \bmod 4, j + nextm(i + xd)] - N[i, j] \\
&M^x[i, j] = |M^x_{brut}[i, j]| \\
&\text{where } M^x_{brut}[i, j] \in [-12, 12], \quad M^x[i, j] \in [0, 11]
\end{aligned} \tag{2.5}$$

- M(s) (i.e. when related to a stratum) has it own way of working, that is defined in the next paragraphs. We focus specifically on M(a), the melodic intervals of the lowest stratum.

[7] If d(p)=x, it means that the following note comes in x beats. So if d=1, M^2[0,0] represents the interval between N[0,0] and N[2,0].

Why is it complicated to consider melodic intervals of strata? Since strata don't have melodic intervals *per se* (they actually do have melodic intervals, but it doesn't really make sense to consider them), we need to redefine what we mean when speaking about the melodic intervals of a stratum. If it is not clear why strata have no inherent melodic intervals, remember that strata are an abstract concept that is used only in mathematical relationships (and respective constraints). People who listen to the music hear the different parts (be they different tessitura, different instruments, ...) and the way these parts interact together in melodic movements and harmonic convergences, rendering a beautiful music, or not. Strata are an abstraction of the harmonic interactions between the parts, and because of this, they are a consequence of the parts: they exist because the parts exist, and not the other way round! And since they are defined according to harmonic principles (as was suggested before, they are successions of vertical alignments), speaking about the proper *melodic* intervals of a stratum makes no sense. One could then conclude that melodic intervals do not apply to strata, and go ahead. Nevertheless, Fux *does* speak about computing the motions between a part and the lowest stratum. And to be able to compute motions, one needs to compare two different melodic intervals. So we need to have a definition for the melodic intervals of a stratum.

Definition of M(s), the melodic intervals of a stratum To understand how we arrive at a definition for the melodic intervals of a layer, we need to remember that the lowest layer is just the collection of all the lowest-sounding notes in the composition. It is therefore quite logical to think of the melodic intervals of the lowest layer as the melodic intervals that lead to all those lowest-sounding notes. We thus define the melodic intervals of the lowest stratum to be: the interval that lead to the note of the lowest stratum in the corresponding part. Let's make this clearer with an example. Let's consider that the lowest stratum consists of the notes [C_{cp_1}, E_{cf}, G_{cp_2}] (where C_{cp_1} indicates that the C belongs to the first counterpoint), that in the *cantus firmus* the interval that lead to the E is a +0 (i.e. staying on the same note), and that in the second counterpoint the interval that lead to the G was a -4 (getting down of two tones). The corresponding melodic intervals array of the lowest stratum would then be [+0, -2]. This example has been written again in a more visual way in equation 2.6 to make it easier to understand. To the left of the equation is the pitch array of each voice mentioned. To the right of the equation is the melodic interval array of each voice mentioned. The numbers in bold red are those corresponding to the lowest stratum.

$$\begin{aligned} N(cf) &= [64, \quad \mathbf{64}, \quad 71] & M_{brut}(cf) &= [\mathbf{+0}, \quad +7] \\ N(cp_1) &= [\mathbf{60}, \quad 67, \quad 74] & M_{brut}(cp_2) &= [+7, \quad +7] \\ N(cp_2) &= [72, \quad 71, \quad \mathbf{67}] & M_{brut}(cp_2) &= [-1, \quad \mathbf{-4}] \end{aligned} \tag{2.6}$$

$$N(a) = [\mathbf{60}, \quad \mathbf{64}, \quad \mathbf{67}] \qquad M_{brut}(a) = [\mathbf{+0}, \quad \mathbf{-4}]$$

The formal definition of the melodic intervals of the lowest stratum is hence as follows: the melodic interval in measure j of the lowest stratum is equal to the last melodic interval in measure j of the part that is the lowest stratum in measure $j + 1$. Remember that this complex definition is needed in order for the computation of the motions to work fine, and that the motions of the lowest stratum, juste as the lowest stratum, are an abstract notion that serves only in formulas and constraints. The motions of the lowest stratum *do not intend to represent any concrete motion really happening in the composition, nor does it correspond to the melodic intervals between the pitches of the lowest stratum.*It may be worth noting that if a part is the lowest stratum all the time,

then the motions of the lowest stratum will be completely equal to those of that part.

$$\forall j \in [0, m-1) :$$
$$M_{brut}(a)[j] = \begin{cases} M_{brut}(cf)[0][j] & \text{if } A(cf)[j+1] \\ M_{brut}(cp_1)[\max(\mathcal{B}(cp_1))][j] & \text{if } A(cp_1)[j+1] \\ M_{brut}(cp_2)[\max(\mathcal{B}(cp_1))][j] & \text{if } A(cp_2)[j+1] \end{cases} \tag{2.7}$$

It might be helpful to have a look at figures 2.8 and 2.9 to understand better how the melodic intervals arrays for the lowest stratum. The melodic intervals of the lowest stratum are those that lead to the notes of the lowest stratum.

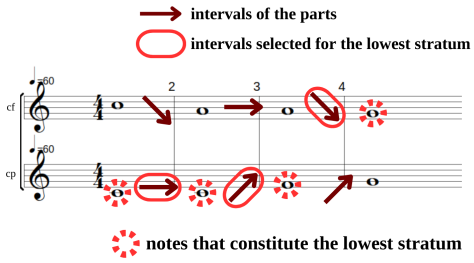

Figure 2.8: Understanding the melodic intervals of the lowest stratum with a first species counterpoint

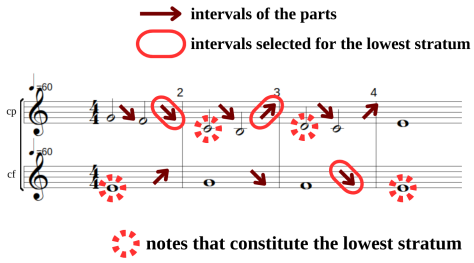

Figure 2.9: Understanding the melodic intervals of the lowest stratum with a second species counterpoint

P(p) `motions`

The motions array represents the motions[8] of a voice v with respect to the lowest stratum. The change from the previous work (where the motions array represented the motions with respect to the *cantus firmus*) was made since Fux considers that the motions should be considered between each voice and the lowest voice. Of course, to be able to compute the motions between two voices, we must compare their melodic intervals, hence, we must deal with melodic intervals of a stratum. This is not a problem anymore since we have defined what the melodic intervals of the lowest stratum mean in the previous sub-section. However, a problem arises when computing the motions of the part that is also the lowest stratum in some measures. When this happens, we end up calculating motions between a part and itself. Any part is inevitably moving in direct motion with itself, and this situation leads to only direct motions being calculated. This becomes problematic when considering costs (it is bad to have direct motions, but it obviously should not be bad to be the lowest stratum), and when considering some constraints. To tackle this problem, the motions of a part are now equal to -1 when the part is also the lowest stratum (which is denoted A(p), see section 2.3.3).

[8]Reminder: there are three types of motion: direct, when both voices move together, contrary, when one voice moves up and the other moves down, and oblique, when one voice doesn't move and the other does

$$\forall p \in \{cf, cp_1, cp_2\}, \quad \forall x \in \{1, 2\}, \quad \forall i \in B, \quad \forall j \in [0, m-1), \quad x := b - i$$

$$motion(p)[i,j] = \begin{cases} 0 & \text{if } (M^x_{brut}(p)[i,j] > 0 > M(a)_{brut}[j]) \\ & \qquad \vee (M^x_{brut}(p)[i,j] < 0 < M(a)_{brut}[j]) \\ 1 & \text{if } M^x_{brut}(p)[i,j] = 0 \oplus M(a)_{brut}[j] = 0 \\ 2 & \text{if } (M^x_{brut}(p)[i,j] > 0 \wedge M(a)_{brut}[j] > 0) \\ & \qquad \vee (M^x_{brut}(p)[i,j] < 0 \wedge M(a)_{brut}[j] < 0) \\ & \qquad \vee (M^x_{brut}(p)[i,j] = 0 = M(a)_{brut}[j]) \end{cases}$$

$$P(p)[i,j] = \begin{cases} -1 & \text{if } A(p)[j] \\ motion(p)[i,j] & \text{if } \neg A(p)[j] \end{cases} \tag{2.8}$$

This equation 2.8 may seem daunting, but it's actually very simple (just a little verbose). It works like this:

For each beat in the composition:

- If a part is also the lowest stratum, P is -1 (i.e. non applicable, otherwise we would calculate the motion between the part and itself)
- If the part moves in the opposite direction to the lowest stratum, P is 0.
- If the part stays where it is and the lowest stratum moves (or vice versa), P is 1.
- If the part moves in the same direction as the lowest stratum, P is 2.

A `is-lowest`

A is an array of boolean variables with a size of m, where each variable indicates whether the corresponding part is the lowest stratum. In other words, $A(v)$ is true if v is the lowest stratum. The notation "A" was chosen as the uppercase of "a", which itself represents the lowest stratum. It is also worth to be noted that only one of the parts can be the lowest stratum at the time. This does not mean that two parts cannot equal the lowest stratum at the same time, it *is* indeed possible that two parts blend in unison in the final chord, and that both pitches are the lowest sounding notes. It means that only one of those two is going to be considered to *be* the lowest stratum (and the other one will be the intermediate stratum). This is needed in order for motions to work well. See 2.1 for the details.

Here is the mathematical definition of the A array:

$$\forall j \in [0, m-1):$$

$$A(cf)[j] = \begin{cases} \top & \text{if } N(cf)[0,j] = N(a)[0,j] \\ \bot & \text{else} \end{cases}$$

$$A(cp_1)[j] = \begin{cases} \top & \text{if } (N(cp_1)[0,j] = N(a)[0,j]) \wedge \neg A(cf)[j] \\ \bot & \text{else} \end{cases} \tag{2.9}$$

$$A(cp_2)[j] = \begin{cases} \top & \text{if } \neg A(cf) \wedge \neg A(cp_1) \\ \bot & \text{else} \end{cases}$$

As can be seen in these equations, only the downbeat of each measure is taken into account when computing the A array. The reason for this is that it is the downbeat note that determines which chord will be *the* chord of the measure, and the other beats are just fioritures. Another reason for this is also that it is only going to serve in contexts where the first note of the measure is relevant.

In practice, there is only an `is-not-bass` array in the code (which is then equal to $\neg A$), as it is almost always more useful to know if a part is *not* the lowest stratum than knowing if it is the lowest one.

Chapter 3

Formalising Fux rules for three-part composition

The purpose of this section is to extract all the rules that Fux mentions in his work and to make sure that they are unambiguous.

It consists of seven subsections: first, a section for the implicit rules (the rules that Fux doesn't mention, but are present in all his examples), then a section for each species, and finally a short section that considers the interaction between different species. There is no section on rules for all species, yet there *are* rules that apply to all species. In fact, all the rules mentioned in the first species section also apply to the other species. Although Fux doesn't explicitly mention this point, it becomes clear when you look at how he teaches and applies these rules in composition.Therefore, the rules for all species are in the first species section. The reason these rules are in the first type section is because Fux himself mentions these rules in his chapter on first type counterpoint.

Each species subsection is divided into two parts: the first part is about setting up Fux's rules and discussing them. The second part is about translating the rules into formal logic.

Some important notes

- **Concerning the green dot** — Please note that all the rules from T. Wafflard's thesis still apply to three-part compositions. The numbering of the rules in this work is the same as in T. Wafflard's work. If a rule is defined with the same number as an existing rule for two-part compositions, this means that the corresponding rule from T. Wafflard's work does not apply to three-part compositions, and that the new rule should be used instead. To make this clearer, there is a green dot (•) next to the rules from T. Wafflard's thesis that got redefined when used in a three-voice composition.

- **Concerning the `[PREF]` marker** — As discussed earlier (see Section 1.1.1), some rules are mandatory and must be followed to find a valid solution, and other rules are just preferences that can be followed to find a *better* solution. The preferences have been marked `'[PREF]'` throughout the chapter to make it clearer which rule is mandatory and which is a preference.

- **Concerning the default costs** — Each time a cost is mentioned in the formalisation rules, the value corresponding to it is its default value, as defined in Appendix B. Note, however, that in practice the value for each cost can be changed by the user to suit their needs.

- **Concerning the purpose of all the rules** — It may be interesting to remember the purpose of these rules. Some rules are concerned with musicality: composing counterpoint that sounds nice; for example, the rule that forbids dissonance. Some rules are concerned with singability: composing counterpoint that

is not too difficult for the human voice to sing; for example, the rule forbidding a melodic leap greater than a sixth.

3.1 Implicit rules

These implicit rules apply to all types of counterpoint. They are never explicitly defined by Fux, but are derived from his many examples. These implicit rules are therefore actually used by Fux, even though he doesn't talk about them.

3.1.1 Formalisation in English

1.H2 and 1.H3 • *First and last notes have not to be perfect consonances anymore.*

Fux doesn't state this in his text, but in many of his examples, we see that when he composes with three voices (and more), the first and last harmonic intervals between the parts and the lowest stratum are not necessary a perfect consonance anymore.

1.H7 and 1.H8 • *The harmonic interval of the penultimate measure must be either a minor third, a perfect fifth, a major sixth, or an octave.*

In two-part composition rules, Fux said that the last harmonic interval had to be either a minor or a majord third (see rule **1.H7** for two voices). This is something he doesn't respect at all in three-part composition, but we can see that he still tries to use minor third and major sixths when possible, in the penultimate measure. The rule from two-part species was thus rewritten in order to be appropriate: either use a perfect consonance, a minor third, or a major sixth.

G8 *The last chord must be composed only of the notes of the harmonic triad.*

Again, this isn't stated explicitly, but we see that all of his examples end with a chord containing exclusively the notes of the harmonic triad.

G9 *The last chord must have the same fundamental as the one of the scale used throughout the composition.*

This rule emanates from an observation of Fux's examples throughout the chapter. The last chord of all his compositions always have the same fundamental as the fundamental of the scale used throughout the composition. When the *cantus firmus* is the lowest stratum, this is not a problem, as the *cantus firmi* always end with the fundamental note of the scale. But when not, it has to be imposed by a constraint, or we may end up with surprising results.

3.1.2 Formalisation into constraints

1.H2 and 1.H3 • *First and last notes have not to be perfect consonances anymore.*

There is no constraint associated with this rule, as it is a relaxation of a rule from the two-part composition rule set.

1.H7 and 1.H8 • *The harmonic interval of the penultimate measure must be either a minor third, a perfect fifth, a major sixth, or an octave.*

$$H[0, m-1] \in \{0, 3, 7, 9\} \tag{3.1}$$

G8 *The last chord must be composed only of the notes of the harmonic triad.*

$$\forall s \in \{b, c\} : H(s)[0, m-1] \in Cons_{h\_triad} \tag{3.2}$$

G9 *The last chord must have the same fundamental as the one of the scale used throughout the composition.*

Since the fundamental of the scale is defined by being the first note of the *cantus firmus*, we impose that the last note of the lowest stratum must be equal to the first one of the *cantus firmus* (taking the modulos into account).

$$N(a)[0, m-1] \mod 12 = N(cf)[0, 0] \mod 12 \tag{3.3}$$

3.2 First species

This section deals with the rules that apply to the first species of counterpoint. As mentioned earlier, these rules apply to *all* species in the context of a three-part composition. In other words, these rules apply whenever $species(p) \in \{1, 2, 3, 4, 5\}$. More specifically, the rules in this section apply to the first beat of each species, except for the fourth species, where they apply to the third beat (see the note on the fourth species 2.3.2).

The first species consists of whole notes only. It is the basis of counterpoint and its simplest case.

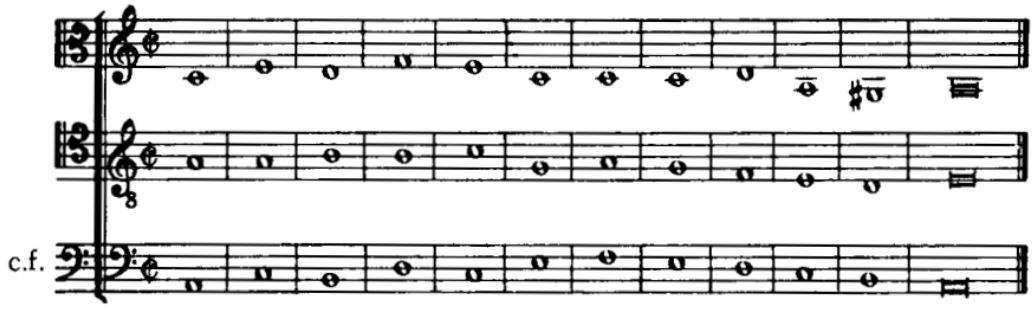

Figure 3.1: Example of a first species counterpoint in three-part composition

3.2.1 Formalisation into English

Structural constraints

1.S1 *All notes are whole notes.*

> "This species consists of three whole notes in each instance." Mann [22, p.71]

This pretty straightforward rule is the very definition of the first species. It adds nothing in comparison with the rules for the two part comparison. It is hence already implemented by the first species for two voices and does not need any consideration.

Harmonic rules

1.H1 • *All notes on the downbeat are consonant with the notes of the lowest stratum.*

> "This species consists of three notes, the upper two being consonant with the lowest." Mann [22, p.71]

This rule is an update of previous **1.H1** (that previously was saying that *all* intervals must be consonants). Fux states that the upper voices and the lowest one are consonant, and not all voices together.

1.H8 [PREF] *The harmonic triad should be used as much as possible.*

> "The harmonic triad should be employed in every measure if there is no special reason against it." Mann [22, p.71]

As a footnote states it [22, footnote, p.71], Fux refers to the "harmonic triad" as being a chord in this position: 1-3-5 (contrary to what is today understood as a harmonic triad). The rule says it is not obligated, but it is preferred, to use the 1-3-5 chord, considering that 1 is the lowest voice.

1.H9 *One might use sixths or octaves.*

> "Occasionally, one uses a consonance not properly belonging to the triad, namely, a sixth or an octave." Mann [22, p.72]

Here, Fux explains that when it is not possible to have a harmonic triad, you can use sixths or octaves instead. Remember that the sixths or the octaves are calculated from the lowest stratum. Since the rule **1.H1** obligates the use of a perfect consonance (i.e. a third, a fifth, a sixth or an octave), when the harmonic triad cannot be used, it is already naturally replaced by a third or a sixth, because no other intervals are allowed. It is thus not a new rule but a restatement of rule **1.H1**.

1.H10 *Tenths are prohibited in the last chord.*

> "One feels that the degree of perfection and repose which is required of the final chord does not become sufficiently positive with this imperfect consonance [(speaking about a tenth)]." Mann [22, p.77]

When Fux says this, he takes a tenth as an example, but it here understood that the final chord cannot include a tenth (third + octave), nor an eight-teenth (third + two octaves), etc. Nevertheless, "simple" third are considered completely valid.

1.H11 [PREF] *Octaves should be preferred over unisons.*

> "Unison is less harmonious than the octave." Mann [22, p.79]

This rule does not bring anything new, as there is already a rule stating that two parts cannot blend in unison (**1.H5**). If no unison is possible, then the octaves will always be preferred over the unison (since the latter is not possible).

1.H12 *Last chord cannot include a minor third.*

> "The minor third is not capable of giving a sense of conclusion." Mann [22, p.80]

Fux later states that minor modes should not include a third altogether, but that sometimes it is impossible to do without it, so the major third *is* allowed in minor modes.

Melodic rules

1.M3 [PREF] *Steps are preferred to skips.*

> "[Each part] follows the natural order closely." Mann [22, p.73]

After having said this, Fux complements his explication by saying the counterpoints should be "moving gracefully, stepwise without any skip". This is clearly a preference, and has already been covered when implementing the first species for two voices (see **G7**). It can thus be ignored in the scope of this thesis.

1.M4 [PREF] *The notes of each part should be as diverse as possible.*

> "[Each part] follows the principle of variety." Mann [22, p.73]

Fux never clearly defines what he means by the "principle of variety". So we try to define it according to what we can read in his work. The examples he gives are a great help. He first writes an incorrect example and then corrects it, saying that the corrected example is better because it follows the principle of variety more closely. The difference between the two examples is that the number of different pitches has been increased. So we can define the principle of variety as: use as many different notes as possible in a single voice.

The principle of variety is therefore clearly a preference.

1.M5 *Each part should stay in its voice range.*

> "One should not exceed the limits of the five lines without grave necessity." Mann [22, p.79]

Fux says here that each part should stay on the musical staff (Fux's "five lines"). Since every staff can be represented differently according to the clef that is used, this rule could be always true. Obviously, Fux meant the staff corresponding to the voice range (treble clef for a soprano, bass clef for a bass, ...).

This is actually something that is already handled when declaring the $N(p)$ arrays, as they are declared with an upper and lower bound ($ub(p)$ and $lb(p)$), corresponding to their voice range.

1.M6 *Melodic intervals cannot be greater or equal to a sixth.*

> "The skip of a major sixth is prohibited." Mann [22, p.79]

This rule is only a restatement of rule **1.M2**, saying that melodic intervals cannot exceed a minor sixth interval.

Motion rules

1.P1 • [PREF] *Reaching a perfect consonance by direct motion should be avoided.*

> "[Reaching] perfect consonance by direct motion [is allowed if] there is no other possibility." Mann [22, p.77]

This is a new rule as it only applies to three-part composition, but it cancels an already existing rule that used to be applied in two-part composition. The same rule in two-part composition states that it is prohibited to reach a perfect consonance using a direct motion. In three-part composition, this is not prohibited anymore, as not doing it is sometimes impossible, and you may thus derogate from this rule.

1.P4 [PREF] *Successive perfect consonances should be avoided.*

> "The necessity of avoiding the succession of two perfect consonances [...]." Mann [22, p.72]

Fux here implies that there should be no two successive perfect consonances. He does not specify whether this rule applies to all three parts at once (i.e. if there was a consonance at bar X between part 1 and part 2, there cannot be one between part 2 and 3 at bar X+1), or whether it applies to each pair of parts separately. That said, in his example (Fig. 91 of the English version [22]), we can clearly see that there is perfect consonance in every bar (parts 1-3, then 1-2, then 1-3, then 2-3, then 1-2). From this we can deduce that *for each pair of parts* it is forbidden for two perfect consonances to follow each other.

However, a closer look at his examples throughout the book reveals that Fux does not respect this rule at all. To name just a few places where this rule does not apply, let's mention figure 108, in the first three measures, between the bass and the *cantus firmus*; figure 109, in the same place, between the *cantus firmus*and the alto; figure 110, in measures 8 and 9, between the bass and the alto. For this reason, this rule must be considered as a preference rather than an absolute constraint.

1.P5 *Each part starts distant from the lowest stratum.*

> "To allow enough space for the voices to move toward each other by contrary motion, the upper voices begin distant from the bass." Mann [22, p.75]

This preference cannot be made clearer: the voices start distant from the lowest stratum.

1.P6 *It is prohibited that all parts move in the same direction.*

> "All voices ascend[ing] [is] a progression which can hardly be managed without awkwardness resulting." Mann [22, p.76]

What Fux is explaining here is simply that the three parts cannot move in the same direction. In other words, if two voices go up, the last one cannot go up. If two voices go down, the last one cannot go down. And if two voices stand still, the last one must move.

1.P7 *It is prohibited to use successive ascending sixths on a direct upwards motion.*

> "Ascending sixths on the downbeat sound harsh." Mann [22, p.77]

This rule is quite simple and states that if one harmonic interval is a sixth, then the next harmonic interval cannot also be a sixth.

3.2.2 Formalisation into constraints

Structural constraints

1.S1 *All notes are whole notes.*
This rule needs no special constraint, since it is the very definition of the first species to consist only of whole notes.

Harmonic rules

1.H1 • *All notes on the downbeat are consonant with the notes of the lowest stratum.*
The new definition of variable H already captures the change in the rule. This means that the equation of rule **1.H1** stays the same.

1.H8 [PREF] *The harmonic triad should be used as much as possible.*
As this rule is actually a preference and not a mandatory rule, it has been implemented as a cost. If the harmonic triad is used, then the cost is 0. Else, it is 1.

$$\begin{aligned} &\forall j \in [0, m-1): \\ &((H(b)[0,j] \neq 3) \wedge (H(b)[0,j] \neq 4)) \vee (H(c)[0,j] \neq 7) \\ &\iff cost_{prefer-harmonic-triad}[j] = 1 \end{aligned} \tag{3.4}$$

1.H9 *One might use sixths or octaves.* As discussed in **1.H9**, there is no constraint to add for this rule.

1.H10 *Tenths are prohibited in the last chord.*

$$H_{brut}[0, m-1] > 12 \implies H[0, m-1] \notin \{3, 4\} \tag{3.5}$$

If the harmonic interval is bigger than an octave, then you cannot use thirds anymore.

1.H11 [PREF] *Octaves should be preferred over unisons.* As discussed before, there is no constraint to add for this rule.

1.H12 *Last chord cannot include a minor third.*

$$H[0, m-1] \neq 3 \tag{3.6}$$

Melodic rules

1.M3 [PREF] *Steps are preferred to skips.*
As discussed in **1.M3**, there is no constraint to add as it already exists (see **G7**).

1.M4 [PREF] *The notes of each part should be as diverse as possible.*
As it is not explained either if this has to be true for the whole partition or only for two following notes, it has been chosen as an arbitrary seven successive notes to apply the rule on. This means that the solution is penalized if a note in measure X was already present in measures [X-3, X+3]. This amount was chosen because it represents the number of flat notes that exist, pushing for the solver to find a solution that contain all of them.

$$\begin{aligned} &\forall p \in \{cp_1, cp_2\}, \quad \forall j \in [0, m-1), \quad \forall k \in [j+1, \min(j+3, m-1)] : \\ &N(p)[0,j] = N(p)[0,j+k] \iff cost_{variety}[j + m * k] = 1 \end{aligned} \tag{3.7}$$

This rule is very interesting because it is the only *across all* rule that has a scope greater than two bars. While without this rule the solver uses constraints that apply only to one bar (think of harmonic constraints) or sometimes to two bars (think of melodic constraints), this constraint is the first to give the solver some "memory" about

the composition. And although seven bars may seem like a small range, it means that the constraint covers a large part of the composition (when dealing with compositions of 10 to 15 bars, of course).

1.M5 *Each part should stay in its voice range.*
As was discussed before, there is no constraint associated to this rule, as it is already covered by the definition of the voice range.

1.M6 *Melodic intervals cannot be greater or equal to a sixth.* As was said before, no constraint must be implemented for this rule as it is a restatement of rule **1.M2**.

Motion rules

1.P1 • [PREF] *Reaching a perfect consonance by direct motion should be avoided.*
Since there is no way in constraint programming to implement a rule that must be obeyed only if possible other than by using a cost, the initial constraint was rewritten to a new one.

$$\begin{aligned} &\forall j \in [0, m-2) : \\ &P[0,j] = 2 \wedge H[0, j+1] \in Cons_p \\ &\quad \iff cost_{\text{direct\_move\_to\_p\_cons}}[j] = 8 \end{aligned} \tag{3.8}$$

Remember: $P[0, j] = 2$ means that the motion is direct.

1.P4 [PREF] *Successive perfect consonances should be avoided.*
As discussed before, this rule is actually a preference.

$$\begin{aligned} &\forall v_1, v_2 \in \{cf, cp_1, cp_2\}, \quad v_1 \neq v_2, \quad \forall j \in [0, m-2): \\ &(H(v_1, v_2)[0, j] \in Cons_) \wedge (H(v_1, v_2)[0, j+1] \in Cons_p) \\ &\quad \implies Cost_{succ\_p\_cons} = 2 \end{aligned} \tag{3.9}$$

The cost has been set to two according to the cost hierarchy defined in T. Wafflard's thesis (a cost of two is a medium cost), but it is possible for the user to change this cost. The costs are discussed in detail in section 4.2.

1.P5 *Each part starts distant from the lowest stratum.*
This is not a strict rule but an indication to make easier for the composer to have contrary motions. Since this is neither a requirement nor a preference, it can simply be added as a heuristic for the solver. This is discussed in section 4.1.2, on heuristics.

1.P6 *It is prohibited that all parts move in the same direction.*
To prevent this, we need only look at the motions between the parts and the lowest stratum. If one of their motions is contrary, then it is guaranteed that the three voices will not go in the same direction (because at least one is contrary). The same applies if one of the motions is oblique. The problem arises when all the movements are direct, because this would mean that the three voices are going in the same direction. So it was forbidden to have all motions direct at the same time.

$$\begin{aligned} &\forall j \in [0, m-2): \\ &\bigvee_{p \in \{cf, cp_1, cp_2\}} M(p)[0, j] \neq 2 \end{aligned} \tag{3.10}$$

Remember that number "2" represents a direct motion.

1.P7 *It is prohibited to use successive ascending sixths on a direct upwards motion.* Either the harmonic interval is not a sixth in any of both positions, or one of them is not moving up.

$$\begin{aligned}&\forall j \in [1, m-1), \quad \forall v_1, v_2 \in \{cf, cp_1, cp_2\} \text{ where } v_1 \neq v_2, \quad \text{sixth} := \{8, 9\}: \\ &(H(v_1, v_2)[0, j-1] \notin \text{sixth}) \vee (H(v_1, v_2)[0, j] \notin \text{sixth}) \\ &\vee M(v_1)[0, j] > 0 \vee M(v_2)[0, j] > 0\end{aligned} \tag{3.11}$$

3.3 Second species

The second species consists only of half notes. It introduces more dissonance than was possible with the first species.

All the rules in this section apply only when $species(p) = 2$, with p being the part mentioned in the rule. Note that the rules for the first species also apply to counterpoints of the second species.

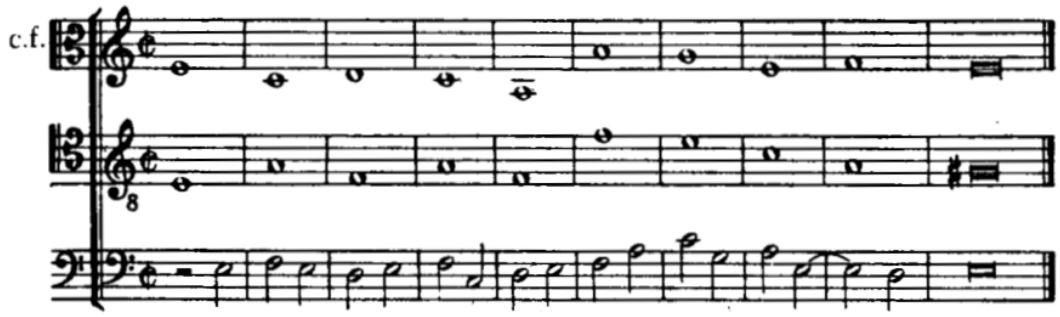

Figure 3.2: Example of a second species counterpoint in three-part composition

3.3.1 Formalisation in English

Harmonic rules

2.H4 *Major thirds are now allowed in the last chord.*

> "A major third [may] appear in the last chord." Mann [22, p.87]

This is a consequence of now using three voices instead of two. Fux makes explicit two implicit rules we had already defined (**1.H2 and 1.H3** and **G8**). It has thus already been implemented in the first species for two voices.

2.H5 *The half notes must be coherent with respect to the whole notes.*

> "The half notes are always concordant with the two whole notes." Mann [22, p.88]

One might ask what Fux meant when he wrote "concordant". Did he mean to say "consonant"? Our take on the question is that he meant that the half notes are written whilst taking the whole notes into account. This interpretation is aligned with the French translation, and even with the Latin original. In other words, Fux just says "there are constraints on the half notes". It is thus not a rule *per se*.

Melodic rules

2.M2 • *It is allowed to ligate the fourth-to-last with the third-to-last or to ligature the third-to-last with the second-to-last.*

> "Ligatures have no place in this species [except] in the final cadence." Mann [22, p.87]

Fux explains that in some cases, you have no other option than ligaturing the fourth-to-last and the third-to-last notes. The reasons he gives for this are all part of the previous mentioned rules (no successive perfect consonances, no unison, ...).

Later on, he also says that the third-to-last and the second-to-last notes can be ligatured (hence producing a whole note).

> "A whole note may occasionally be used in the next to last measure." Mann [22, p.93]

He says that in the chapter about third species, but it seems that this applies even in cases where the second species is not used in combination with the third (see figures 134, 173 and 174 of the English version).

He doesn't state clearly if the three of them can get ligatured, but it seems quite obvious that this is not allowed, as it would introduce a lot of redundancy in the composition. It is hence decided that the rule is: a ligature may happen in one case or in the other, but not in both.

We thus have to relax the already existing constraint from two-part composition stating that no two consecutive notes can be the same, to accept it in some cases.

Motion rules

2.P3 [PREF] *Successive fifths on the downbeat are only allowed when they are separated by a third on the upbeat.*

> "A half note may, for the sake of the harmonic triad, occasionally make a succession of two parallel fifth acceptable - which can be effected by the skip of a third." Mann [22, p.86]

Fux didn't speak about prohibiting two parallel (i.e. consecutive) fifths in the second species for two voices. That being said, it is indeed prohibited in three parts composition as you cannot have two successive perfect consonances (see rule **1.P4**). We thus have to relax constraint **1.P4** in order to accept two successive consonances, when the two successive fifths flank a third. And since the rule on successive perfect consonances is actually a preference, this means that the cost of successive perfect consonances is not applied if those two consonances are fifths and there is a third in between.

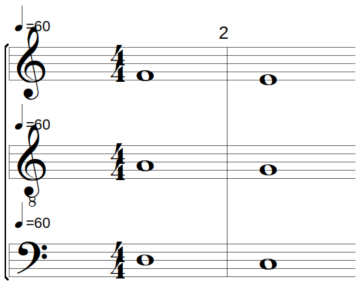

Figure 3.3: Successive fifths - prohibited

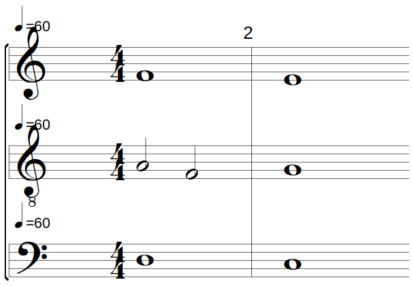

Figure 3.4: Successive fifths separated by a third - valid

3.3.2 Formalisation into constraints

Harmonic rules

2.H4 *Major thirds are now allowed in the last chord.*
No need to add a new constraint as this rule is already covered by rules **1.H2 and 1.H3** and **1.H8**.

2.H5 *The half notes must be coherent with respect to the whole notes.*
No need to add a new constraint as this is not an actual rule.

Melodic rules

2.M2 ● *It is allowed to ligate the fourth-to-last with the third-to-last or to ligature the third-to-last with the second-to-last.*
This is a relaxation of the two-voice rule **2.M2** "two consecutive notes cannot be the same".
The reason why this rule has been implemented a constraint relaxation instead than as a cost is because Fux does not say that ligaturing is bad, he just presents it as a new option offered by the three-part composition.

$$\begin{aligned}&\forall j \in [1, m-1), \quad j \neq m-2 : \\ &((N[2, j-1] \neq N[0, j]) \wedge (N[0, j] \neq \wedge N[2, j])) \\ &\wedge \\ &((N[2, m-3] \neq N[0, m-2]) \vee (N[0, m-2] \neq N[2, m-2]))\end{aligned} \tag{3.12}$$

The first line prohibits the ligatures except where they are allowed, and the second lines states that only one ligature can occur.

Motion rules

2.P3 [PREF] *Successive fifths on the downbeat are only allowed when they are separated by a third on the upbeat.* This rule is a relaxation of the cost **1.P4** defined above, and is thus rewritten to correspond to a special case that occurs with the second species.
In the following equation, only p_1 must be a second species counterpoint, p_2 can

be any species.

$$\forall p_1, p_2 \in \{cf, cp_1, cp_2\} \text{ where } p_1 \neq p_2, \quad \forall j \in [0, m-2) :$$

$$Cost_{succ\_p\_cons} = \begin{cases} 0 & \text{if } (H(p_1,p_2)[0,j] \notin Cons_p) \vee (H(p_1,p_2)[0,j+1] \notin Cons_p) \\ 0 & \text{if } (H(p_1,p_2)[0,j] = 5) \wedge (H(p_1,p_2)[0,j+1] = 5) \\ & \qquad \wedge (H(p_1,p_2)[2,j] = 3) \vee (H(p_1,p_2)[2,j] = 4) \\ 2 & \text{otherwise} \end{cases} \tag{3.13}$$

The meaning of this equation is that $Cost_{succ\_p\_cons}$ is equal to zero if one of the two considered consonances is not perfect (because then we do not have two successive perfect consonances), or if we have two successive fifths with a third in between. Otherwise (when we have perfect consonances), the cost must be set.

3.4 Third species

The second species consists only of quarter notes. It introduces even more dissonance than the second species and opens the space for more variation.

All the rules in this section apply when $species(p) = 3$. Note that the rules for the first species also apply to counterpoints of the third species.

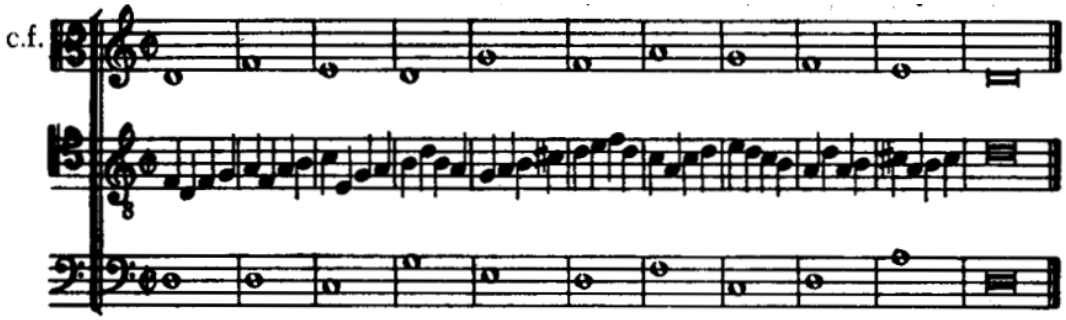

Figure 3.5: Example of a third species counterpoint in three-part composition

3.4.1 Formalisation in English

Harmonic rules

3.H5 *The quarter notes must be coherent with respect to the whole notes.*

> "The quarters have to concur with the whole notes of the other voices." Mann [22, p.91]

When using the word "concur", Fux more than probably means "are put in relation", and not "are consonant". As for rule **2.H5** in the second species, where the half notes had to be *concordant* with the *cantus firmus*. It is not a rule by itself, Fux is only annunciating that some rules should be followed (i.e. the other constraints).

3.H6 `[PREF]` *If the harmonic triad could not be used on the downbeat, it should be used on the second or third beat.*

> "Take care whenever you cannot use the harmonic triad on the first quarter occurring on the upbeat, to use it on the second or third quarters." Mann [22, p.91]

This rule is quite clear and speaks for itself.

Melodic rules

Fux introduces no new melodic constraints for the third species.

Motion rules

Fux introduces no new motion constraints for the third species.

3.4.2 Formalisation into constraints

Harmonic rules

3.H5 *The quarter notes must be coherent with respect to the whole notes.* As has been discussed in the previous section, there is no constraint to add for this rule, which isn't really a rule.

3.H6 [PREF] *If the harmonic triad could not be used on the downbeat, it should be used on the second or third beat.*

This rule is quite clear and speaks for itself. Since this is not a strict rule but an advice, it was treated as a cost.

$$\begin{aligned} &\forall j \in [0, m-1): \\ &(H[1,j] \notin Cons_{h\_triad}) \wedge (H[2,j] \notin Cons_{h\_triad}) \\ &\iff cost_{harmonic-triad-3rd-species}[j] = 1 \end{aligned} \tag{3.14}$$

3.5 Fourth species

As a reminder, the fourth species consists of syncopations. Each note is played on the upbeat and is ligated to the next note on the upbeat.

All the rules in this section apply when $species(p) = 4$. Note that the rules for the first species also apply to counterpoints of the third species.

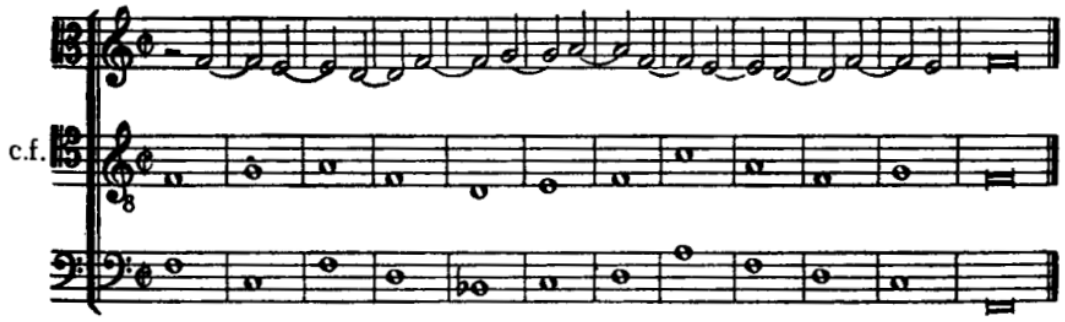

Figure 3.6: Example of a fourth species counterpoint in three-part composition

3.5.1 Formalisation in English

Structural constraints

4.S1 *The fourth species is staggered by two beats.*

> "The ligature is nothing but a delaying of the note following." Mann [22, p.95]

Fux here insists on a fact that we have already discussed in 2.3.2. The fourth species behaves as if its upbeat were the downbeat and its downbeat were the upbeat of the previous measure.

4.S2 *All parts can become the lowest stratum somewhere in the composition.*

> "The tenor takes the place of the bass - a thing that not only the tenor may do, but also the alto and even the soprano." Mann [22, p.100]

Fux speaks here about our concept of strata. The tenor can become the lowest stratum, just like the alto and the soprano may do. This is a fundamental concept of the generalization of Fux counterpoint to three voices, and has already been extensively discussed before (see section 2.1).

Harmonic rules

4.H5 [PREF] *Imperfect consonances are preferred over fifth intervals, which in turn are preferred over octaves.*

> "The fifth is a perfect consonance, the octave a more perfect one, and the unison the most perfect of all; and the more perfect a consonance, the less harmony it has." Mann [22, p.97]

This rule is as clear as it gets.

Melodic rules

Fux introduces no new melodic constraints for the third species.

Motion rules

4.P3 [PREF] *Successive fifths are allowed when using ligatures.*

> "[It would be impossible to remove] the ligatures because of another consideration, the immediate succession of several fifths." Mann [22, p.95]

By saying that it exists a rule that prohibits the succession of fifths (which is actually just a particular case of rule **1.P4**, stating that you cannot have two successive perfect consonances) when there is no ligature, Fux is telling us in an indirect way that this rule is not applicable when there are ligatures. He further complements by saying "there is great power in ligatures - the ability to avoid or improve incorrect passages".

The conclusion is that successive fifths are allowed in the fourth species.

4.P4 [PREF] *Resolving to a fifth is preferred over resolving to an octave.*

> "A dissonance that resolves to a fifth is more acceptable than a dissonance that resolves to an octave." Mann [22, p.98]

This rule could not be clearer.

4.P5 *Stationary movement in the bass implies dissonance in the fourth species part.*

> "If I said that the first note of the ligature must always be consonant, that applies only to the instances in which the lower voice moves from bar to bar, but not the instances in which the bass remains on a pedal point, that is, in the same position. In such a case a ligature involving only dissonances is not only correct but even very beautiful." Mann [22, p.98]

The rule evoked here cancels the previous rule **4.H1W** that stated that all notes should be consonant. From now on, if the lowest stratum has a stationary movement, the corresponding delayed note in the fourth species must be a dissonance, instead of a consonance.

4.P6 *A note provoking a hidden fifth gets replaced by a rest.*

> "Here a hidden succession of fifths occurs, which is easily perceptible to the ear and should be avoided in three part composition. This may be managed by using a rest." Mann [22, p.98]

Fux's uses the term 'hidden fifth' without any prior definition. It is therefore difficult to be sure of what he meant, since the traditional terms for such progressions are as vague and variable as the traditional rules that govern them. Nevertheless, most people seem to agree on the following definition of a 'hidden interval': a hidden fifth or hidden octave is when you approach a perfect fifth or perfect octave by direct motion. [35, p.31]. Looking closely at figures 137, 151 and 152 of the English version of *Gradus ad Parnassum*, this definition is consistent with Fux's interpretation.

The point of the rule then is: if a solution leads to a hidden fifth, then the note that provokes the fifth is replaced by a rest. This rule is an *a posteriori* rule: it applies after the solution has been found. The current rule thus complements the rule **4.P3** (about successive fifths in fourth species) and the rule **1.P1** (about direct moves to perfect consonances) without changing them.

See figures 3.7 and 3.8:

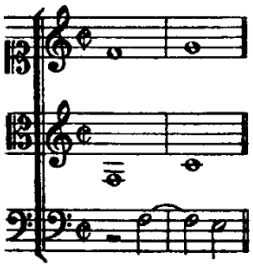

Figure 3.7: Invalid solution featuring hidden fifths

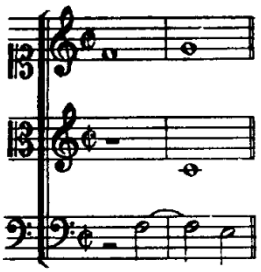

Figure 3.8: Valid solution replacing the hidden fifth by a rest.

3.5.2 Formalisation into constraints

Structural constraints

4.S1 *The fourth species is staggered by two beats.*

There is no constraint to add since this rule is the very definition of the fourth species.

4.S2 *All parts can become the lowest stratum somewhere in the composition.*

Again, there is no constraint to add since this rule is already covered by the concept of lowest stratum.

Harmonic rules

4.H5 [PREF] *Imperfect consonances are preferred over fifth intervals, which in turn are preferred over octaves.*

This rule is almost covered by the existing costs (see **1.H6**), as a perfect consonance has a higher cost than an imperfect consonance. But Fux says not only that imperfect consonance should be preferred over perfect ones, he says that fifths should be preferred over octaves. This precision in the rule (fifth is better than octave) could be solved by either putting a higher cost to octaves and lower one to fifths, or to put the cost for fifth before the cost for octaves in the lexicographical array of costs, but this is discussed in the parts about costs (see 4.2).

Motion rules

4.P3 [PREF] *Successive fifths are allowed when using ligatures.*

The point of this rule is that Fux introduces an exception to **1.P4**: successive fifths are allowed in the fourth species.

We shall then amend the rule **1.P4** (don't forget that it is a cost) to allow successive fifths in any case, and rewrite it as:

$$\forall p_1, p_2 \in \{cf, cp_1, cp_2\}, \quad \text{with } p_1 \neq p_2, \quad \forall j \in [0, m-2): \\ Cost_{succ\_p\_cons} = \begin{cases} 0 & \text{if } (H(p_1,p_2)[0,j] \notin Cons_p) \vee (H(p_1,p_2)[0,j+1] \notin Cons_p) \\ 0 & \text{if } (H(p_1,p_2)[0,j] = 5) \wedge (H(p_1,p_2)[0,j+1] = 5) \\ 2 & \text{otherwise} \end{cases} \tag{3.15}$$

The meaning of this equation is that the cost is set to zero if the two consecutive intervals are not perfect consonances, or if the consecutive intervals are both fifths. Otherwise, the cost must be set.

4.P4 [PREF] *Resolving to a fifth is preferred over resolving to an octave.*

This is already covered by the rule **4.H5** (prefer fifths over octaves), since preferring fifths over octaves in *all* cases implies preferring to resolve to a fifth rather than to an octave.

4.P5 *Stationary movement in the bass implies dissonance in the fourth species part.*

$$\begin{aligned} &\forall j \in [0, m-1): \\ &M(a)[0,j] \neq 0 \iff H[2,j] \in Cons \\ &M(a)[0,j] = 0 \iff H[2,j] \in Dis \end{aligned} \tag{3.16}$$

4.P6 *A note provoking a hidden fifth gets replaced by a rest.*

$$\begin{aligned} &\forall j \in [1, m-1): \\ &H[0,j] = 7 \wedge P[0,j] = 2 \iff N(0,j-1) = \emptyset \end{aligned} \tag{3.17}$$

This rule is very special because it applies after the search, not during the search. More precisely, after the search is started and a result is found, only then does this rule begin to apply. To understand why, remember that the suppressed note is suppressed because it causes a hidden fifth. But the only way to have a hidden fifth is to have an existing note, you cannot have a hidden fifth to a non-existing note. So we don't delete the note during the search, because then we would also delete the hidden fifth

(because you can't have a hidden fifth without a note) and so we would change the solution. You have to find a solution first, and then remove the possible note that causes a hidden fifth.

This is actually a very interesting property, because here we have a note that is not played and yet has an effect on the composition.

3.6 Fifth species

As a reminder, the fifth species is a mix of all previously mentioned species, which means that it combines the rules from all previous species.

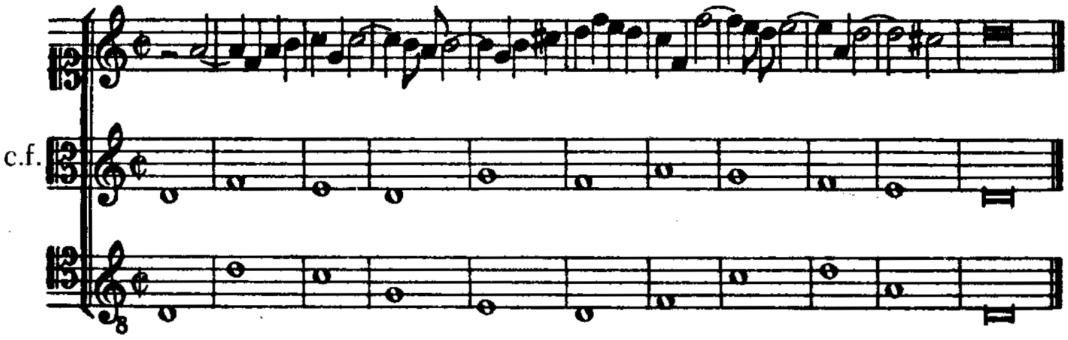

Figure 3.9: Example of a fifth species counterpoint in three-part composition

No additional rules (be they harmonic rules, melodic rules or motion rules) were observed by Fux when writing a three-part counterpoint with the fifth species.

However, in order to have some rhythmic variety when composing with two fifth species counterpoints, it was decided to impose a rule that Fux never mentioned. We know that the fifth is just all the other species combined, and the way the solver understands this is that it considers each note of the composition to belong to a given species. This is represented as S[i,j], where $S[i, j] = x$ means that the i-th note of the j-th measure belongs to the x-th species. This gives us a metric for similarity, since two fifth-species counterpoints that have the same S array will be rhythmically redundant. To tackle this problem, we force the solver to find a solution where $S(cp_1)$ must be at least half as different as $S(cp_2)$. For more information on this notation and the way the solver works with the fifth species, see section 7.2.2 of T. Wafflard's thesis, where the implications of the S array are fully developed.

As can be seen in Figure 3.10, the first composition is indeed rhythmically redundant, since both fifth-species counterpoints have (almost) the same rhythm. The second is much less redundant, since it was required that the counterpoints use different species for at least 50% of the beats.

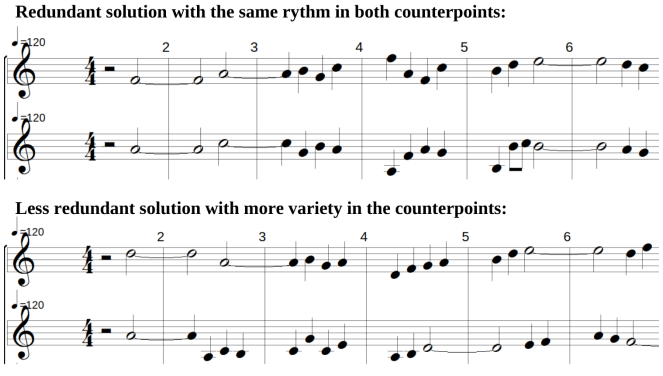

Figure 3.10: Two compositions of two fifth-species counterpoint, one redundant and the other not

$$\sum_{i=0}^{3}\sum_{j=0}^{m-1}(S(cp_1)[i,j] = S(cp_2)[i,j]) < \frac{s_m}{2} \tag{3.18}$$

This equation is only true if both counterpoints are of the fifth species, and its meaning is that the sum of the times $S(cp_1) = S(cp_2)$ must be less than half the number of notes in the composition.

3.7 Writing a three-part composition using various species

In *Gradus ad Parnassum*, Fux almost always writes his counterpoints using a combination of the first species and another species, i.e. either a counterpoint of the first species and another of the first species, a counterpoint of the first species and another of the second species, and so on, up to a counterpoint of the first species and another of the fifth species. However, he sometimes creates other combinations, either with two of the same species (for example, two of the fifth species) or with two different species (for example, the second and the third).

When creating combinations of different species, Fux doesn't give any specific rules for combining them. It seems that the different species interact with each other as they would do with the first species, taking into account only the first beat of the measure. For example, if we compute a first counterpoint belonging to the third species and a second counterpoint belonging to the second species, the rules that apply between the two counterpoints will be set between all the beats of the first counterpoint and the first beat of the second counterpoint and between all the beats of the second counterpoint and the first beat of the first counterpoint. This corresponds to what we would have done if we had applied the rules between a counterpoint and the *cantus firmus*, where the rules apply between all beats of the *cantus firmus*and the first beat of the counterpoint.

Chapter 4

Searching for the best existing solution

Three-part composition is much richer than two-part composition. In fact, it offers many more possibilities than the two-part composition. In constraint solver terms, this means that the search space is much larger, which means two things: the computational complexity of finding a solution is greater, and there are many more possible solutions. The first difficulty is to find a valid solution in this large search space, and the second difficulty is to find the solution that best respects the preferences expressed by Fux.

In this chapter, we first discuss the computational aspect and then turn to the preference management aspect. The computational complexity section covers: the search algorithm used, the heuristics implemented, and some of the factors that influence the speed of finding a solution; whereas the preference management section covers: discussing how to translate Fux's preferences into solver costs, and comparing the different methods that exist for doing so.

4.1 Dealing with the higher computational complexity

As we have just said, composing a three-part counterpoint is more computationally demanding than composing a two-part counterpoint. The search space has been greatly expanded by adding a whole new set of variables, and the time it takes to find a solution may be too high if you do not think about optimising the search. In addition, adding a third voice to a composition does not bring many new constraints (which would help to discard some potential solutions faster), but it does bring many preferences, which in constraint programming are translated into costs. The solver must be efficient in finding the solution that has the lowest cost (i.e. that is best in terms of musicality).

4.1.1 Using Branch-And-Bound as a search algorithm

To cope with the increased complexity brought about by the three-part composition, it was decided to switch from the Depth First Search algorithm (used in T. Wafflard's thesis) to a more efficient Branch and Bound (BAB). This allows us to handle costs properly and to find faster solutions. Moreover, the BAB algorithm can also produce non-optimal results, which is very valuable since finding the best overall solution can be time-consuming. When starting the search for a solution, it is now possible to ask for the next solution (i.e. a better solution than the one found previously, and if none was found previously, then just any valid solution), or for the best solution. In the latter case, the solver will continue to search until it finds the best solution or until it is stopped, returning a better solution each time it finds one.

4.1.2 Heuristics

Main heuristics

When it comes to finding a solution, we obviously need some heuristics to guide the search, because there are so many different possibilities for a three-part composition. To know which heuristics to use, simply think about the most important variable to fix first. In case it is not clear enough, the key to writing counterpoint for many voices is to know what the bass is doing. This is true whether the composer is a human or a constraint solver. So the first heuristic follows naturally, and it is: branch on the lowest stratum array, take the highest constrained variable yet, and try with any value less than the median. This value branching was thought of because the possible values for the lowest stratum are those of the notes of the three parts. Since the lowest stratum is designed to always be equal to the lowest sounding note, it doesn't make sense to try to give it a high value. Therefore, lower values are chosen.

The other central heuristic is the one that instructs the solver to branch on the variables of the N arrays (containing the pitches of the voices), choosing as a priority the variables whose domain is small. This choice is motivated by the fact that in the case of a highly constrained counterpoint (5th species) and a weakly constrained counterpoint (1st species), the counterpoint should not only seek to improve the highly constrained counterpoint, but also the weakly constrained counterpoint. This is why the heuristic chosen is based on the size of the domain and not on the level of constraint. The choice of the value of the variable is then random. This ensures maximum diversity in the final composition and quickly leads to a solution whose notes are varied.

Additional heuristics

A first additional heuristic is to branch to the array representing the species contained in the fifth species, to ensure a varied composition. If we are dealing with a counterpoint of the fourth or fifth species, we also branch on the "no ligature cost", so that the solver explores solutions in which the notes are linked, since this is the very nature of the fourth species (both when it is used "pure" and when it is used within the fifth species).

The rule **1.P5** states that the voices should start distant, and as suggested in the section on rules, this should be implemented in a heuristic. However, when we implemented the heuristic that all voices should start distant from the lowest one, we did not see any improvement, neither in search speed nor in solution quality. In fact, it sometimes slowed down the search, so this heuristic was dropped. Furthermore, Fux's advice that the voices should start far apart in order to progress in the opposite direction is only true if the bottom layer moves up. If the bottom stratum moves down, the top strata should move up, so starting far apart becomes a compositional disadvantage in this case (as the voices are limited by their range).

4.1.3 Time to find a solution

Many factors come into play in determining how long it will take the solver to find a valid solution. These factors are mainly: the species of counterpoints, and the spacing between the voice ranges of the counterpoints.

In general, the solver is able to find a valid solution fairly quickly (in the order of a second). However, in some cases it is more difficult to find a valid solution and the solver may take a little longer to find one. These cases are generally related to the three factors mentioned above, and we will discuss them a little. Each of these factors makes

the search a little more difficult. A single factor may have no effect on the search time, and sometimes it is at the intersection of the factors that an effect is discovered.

- **Species of the counterpoint** – The more complex the species of the counterpoints (think of the 3rd and 5th species, for example), the longer it will probably take the solver to find a solution. The reason is quite obvious: the more complex the species, the more variables the solver has to play with and the greater the range of possibilities.
- **Distance between the voice ranges** – The closer the ranges of the voices, the greater the risk that the solver will take a long time to find a solution. For example, finding a solution by giving the two counterpoints the same voice range as the *cantus firmus* is more time-consuming than selecting distant voice ranges. This is because the voices cannot form a unison, and the possibilities for each voice are therefore smaller when their range is close together.

There are also cases where the exact combination of two given vocal ranges for two given species with a given mode does not give a solution, but by changing the vocal range a little the solver finds a solution immediately, quite surprisingly. It is still unclear why this happens. Our best guess is that there are some combinations of parameters for which the solver has difficulty finding a solution, given the very large number of constraints that apply to the voices. However, this doesn't happen very often.

It is worth noting that the solver's greatest difficulty (in all cases) is finding a valid solution. Once a valid solution has been found, the solver quickly finds a whole series of solutions, each one better than the previous one (until, of course, it is difficult to find the best solution: then the solver starts taking some time again). The distance between the best solution and the solution found by the solver before reaching a plateau in the search is difficult to estimate. However, it should be remembered that the tool is intended to be used iteratively (the user tries a combination, changes the cost, tries again, etc.), and that in the end it is the user's human preferences that count, and there is no guarantee that the best solution will be the one the solver thinks is best. This means that it is the direction the solver takes that is important, not the best solution.

4.2 Designing the costs of the solver to be as faithful as possible to the preferences of Fux

A constraint solver cannot say whether one valid solution is better than another valid solution, and yet we need it to produce the best possible musical solution. We can already say that one solution is better than another if it respects Fux's preferences (see 1.1.1) more. In order for the solver to understand these preferences and to be able to distinguish the musical quality of two solutions, we give costs to it. The lower the cost, the higher the preference.

Knowing that we are looking for the solution whose cost must be as low as possible, the question arises: how can we calculate the cost in order to best reflect the preferences expressed in *Gradus ad Parnassum*?

The way to translate each preference into a corresponding cost has of course been formalised in the previous sections, but that's not the crux of the matter. The question we face here is: what is the best way to combine all these individual costs to get the most accurate result in terms of what Fux is trying to convey in terms of preferences?

Three main ways of doing this have been identified. These are, in order from the simplest to the most complex:

1. a linear combination of all costs,

2. computing the maximum over all costs and minimising it,

3. a lexicographic order search,

We first describe each of these techniques and their respective advantages, and then compare them (and the results they produce).

Throughout this section we talk about some specific costs. All these costs are listed in 4.2.3 with a short explanation. Some more details can be found in B.3, and for the full description of a specific cost, please refer to its corresponding rule.

4.2.1 Linear Combination

The first method of calculating our costs is a linear combination. This is the technique used in T. Wafflard's thesis. More precisely, it uses a linear combination in which all the weights are equal to one.

To be more precise about the method used to calculate the total cost in T. Wafflard's thesis, here is a more detailed explanation: there exists a total cost, τ, which is equal to the sum of all individual costs, $\mathcal{C}$. The next step is to minimise τ. Each $\mathcal{C}_i$ is usually itself a sum of sub-costs. Take, for example, the cost of motions, $\mathcal{C}_{motions} = \sum_j P_{costs}[j]$. This cost is the sum of all sub-costs of the motions (one per motion): by default, a contrary motion has a sub-cost of 0, an oblique motion has a sub-cost of 1 and a direct motion has a sub-cost of 2. These default values can be changed by the user to be set somewhere on a scale that ranges from 0 to $64m$. For example, the user could set the oblique motion cost to be equal to 0, and the cost for direct and contrary motion to be equal to $64m$, in order to get a composition filled with as many oblique motions as possible (always in accordance with the basic rules from *Gradus ad Parnassum*, i.e. all voices are never going to go in the same direction, see **1.P6**).

As mentioned at the beginning of this subsection, this procedure can be understood as a linear combination with weights of one only. However, since the cost factors are given different values according to the user's choices, this method is actually more like a regular linear combination, except that the weights are not multiplied by the costs once the latter have been set, but the costs are themselves made larger or smaller before the linear combination is calculated.

The linear combination has two major advantages: ease of implementation and high comprehensibility. However, it has a major drawback: since the total cost τ that gets minimised is the sum of all costs $\mathcal{C}$, the best solution might be a solution where one cost is absolutely huge and all the others are small. This might not be a problem if the outstanding cost is not really relevant, but if it is the cost of not using a harmonic triad, it goes completely against the preferences that Fux conveys in his work, making the solution inappropriate. A representation of this situation can be found in the figure 4.1.

Another disadvantage of linear combination is that the result is quite unpredictable: changing the value of the cost may or may not make a difference, and you may need to set huge values to see a real effect. For example, if a composer really wants oblique motion, they may be forced to set the cost of the other types of motion to a huge value, or they may not see the difference between the default solution and their personalised solution. This is due to the fact that all the costs are mixed together and form an indistinguishable soup that the solver considers as a whole, and a small increase in the cost of the direct and contrary motions is very likely to be absorbed into this soup without any change being noticed.

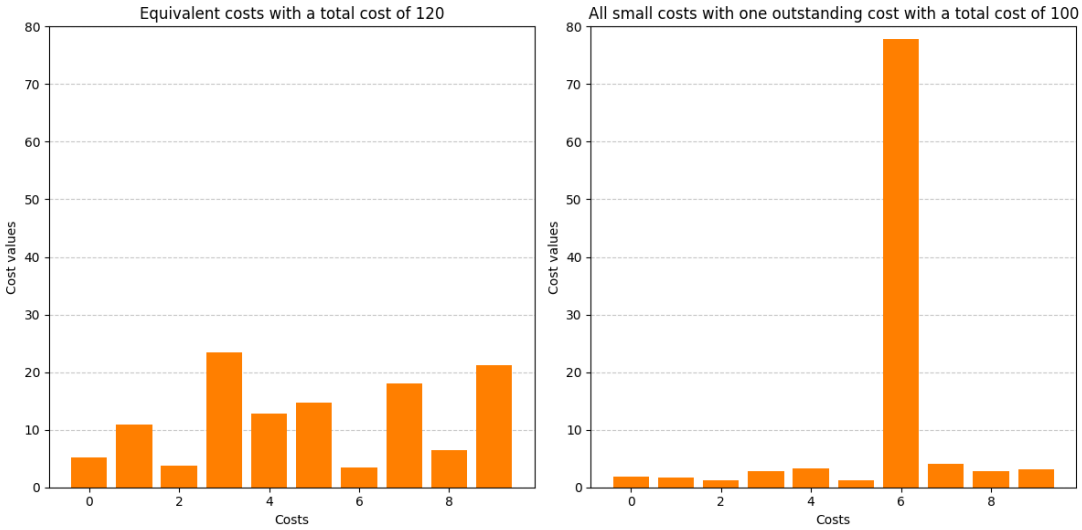

Figure 4.1: Example of a situation where a solution with an outstanding cost is preferred to a solution with equivalent low costs when using a linear combination

These two drawbacks make the linear combination solution for the costs hardly acceptable when it comes to representing the preferences. We therefore examine the other two options for adjusting costs.

4.2.2 Minimising the maxima

In order to overcome the problem of outstanding costs that we encountered when considering the linear combination solution, one could consider a technique that specifically addresses these outstanding costs: namely, minimising the maximum cost.For example, τ the total cost, could be set equal to the current largest cost. By doing this, the solver would try to find a solution where the focus is on the worst cost and try to reduce it before trying to reduce the other costs.

The problem with this method arises when one cost is significantly higher than the others because it has been defined that way. Let's go back to our example of the composer wanting as many oblique motions as possible. You will set the cost for direct motion and contrary motion to the highest possible cost and start the search. As we've already discussed, it is not possible to have only oblique motions, since this would contradict the rule that not all voices can move in the same direction (**1.P6**). As a result, there will always be contrary motions, and since the cost for them has been set very high, it would be impossible for the solver to converge to a good solution. This creates a bottleneck effect, where once the solver has reached the best potential value of the worst cost, it cannot continue to find better solutions.

Furthermore, even when considering a less extreme case (e.g. the default setting), this method requires a normalisation of the costs: there are $3 \times (m-2)$ sub-costs for the variety cost, $3 \times (m-1)$ sub-costs for the motion cost, but only m sub-costs for the octave cost. This means that without normalisation, the motion cost will be on average three times larger than the octave cost, which means that the solver will put three times more effort into minimising the motion cost than the octave cost, which is unfair and unpractical.

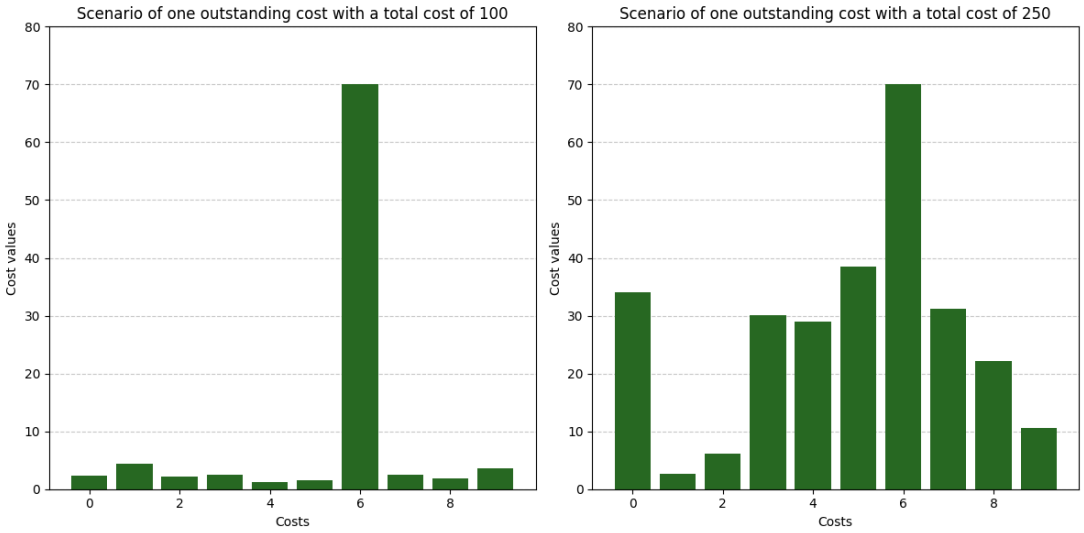

Figure 4.2: Example of two situations where a cost causes a bottleneck in the search because the solver cannot distinguish between left and right situations. The solver will blindly choose one of the two solutions, even if the solution on the left is obviously better.

4.2.3 Lexicographic Order

The second way of dealing with the costs is to arrange them in an array and then perform a lexicographic minimisation. In other words, the costs would be arranged in order of importance: from most important to least important. The most important cost to minimise would be placed first in this array, and the solver would only try to minimise the other costs if the first cost remained the same or decreased. This method makes a lot of sense when you think about the rules that emanate of *Gradus ad Parnassum*. For example, Fux says that perfect consonance can be achieved by direct motion if there is no other possibility. This means that, all other things being equal, we would prefer to achieve perfect consonance by oblique or contrary motion, but that between a bad solution (respecting almost no preferences) in which perfect consonance is not achieved by direct motion, and a good solution (respecting almost all preferences) in which perfect consonance is achieved by direct motion, we would choose the good solution.

Some costs are also more important than others in absolute terms. For example, when Fux says that an imperfect consonance is preferred to a fifth, which is preferred to an octave. This amounts to lexicographicly ranking the cost of using an octave first (because we really don't want octaves), and then the cost of using a fifth (and there is no cost of using an imperfect consonance, since Fux indicates that this is preferable).

$$\tau = [\underbrace{\mathcal{C}_{\text{octaves}}}_{\text{minimise this first}}, \mathcal{C}_{\text{fifths}}] \tag{4.1}$$

Figure 4.3: Array of costs demonstrating the practicality of a lexicographic order solving.

A second example, which ties in particularly well with the first, is that Fux tells us that the harmonic triad must be used in every measure unless a rule forbids it. In saying this, he places the preference for the harmonic triad above all other preferences, because the only reason that can prevent the use of a harmonic triad is a fixed con-

straint (and not a preference). You'll notice that the harmonic triad consists of a fifth (which is a perfect consonant), so Fux is telling us that we'd rather use a fifth in a harmonic triad than an imperfect consonant outside a harmonic triad. The lexicographic order search is the only one that allows this kind of concept to be taken into account, because in a linear combination these two preferences would be mutually "exclusive"[1]: the first preference would add a cost where the second preference would not, and the second preference would add a cost where the first would not.

$$\tau = [\underbrace{\mathcal{C}_{\text{harmonic\_triad}}}_{\text{minimise this first}}, \underbrace{\mathcal{C}_{\text{octaves}}}_{\substack{\text{and start minimizing this only if it is not possible anymore to minimise the harmonic triad cost}}}, \mathcal{C}_{\text{fifths}}] \tag{4.2}$$

Figure 4.4: Array of costs demonstrating the practicality of a lexicographic order solving.

And in this way we can keep integrating the different costs until we get a full array τ with all the costs ordered in a lexicographic way.

Of course, it is not always as simple as in the examples above, because it is not always easy to determine which cost has priority over which other. Sometimes Fux is very clear about it (e.g. for the harmonic triad cost, which Fux says has priority over everything else), and sometimes he isn't (do we prefer no off-key notes, or as much variety as possible?) This is a drawback of this method, because we have to hierarchise the costs, even if the choice is difficult. What's more, once the costs are ranked, their order becomes absolute and the solver loses some of its flexibility.

Knowing this, we came up with a suggested order that should be as close as possible to Fux's preferred order (or at least what we understood him to convey as his preferred order in *Gradus ad Parnassum*). This order should of course be changeable at the composer's discretion[2]. The default order we have agreed upon is as follows. Please note that where a cost is followed by a number in brackets, this means that it only applies if the corresponding species is used.

1. $\mathcal{C}_{\text{no\_syncope}}$[3] [4, 5]
2. $\mathcal{C}_{\text{successive\_p\_cons}}$
3. $\mathcal{C}_{\text{harmonic\_triad}}$
4. $\mathcal{C}_{\text{harmonic\_triad\_3rd\_species}}$ [3]
5. $\mathcal{C}_{\text{octaves}}$
6. $\mathcal{C}_{\text{penult\_thesis\_is\_fifth}}$[4] [2]
7. $\mathcal{C}_{\text{fifths}}$
8. $\mathcal{C}_{\text{off\_key}}$[5]
9. $\mathcal{C}_{\text{variety}}$
10. $\mathcal{C}_{\text{m2\_eq\_zero}}$[6] [3, 4, 5]
11. $\mathcal{C}_{\text{not\_cambiata}}$[7] [3, 5]
12. $\mathcal{C}_{\text{motions}}$
13. $\mathcal{C}_{\text{m\_degrees}}$[8]
14. $\mathcal{C}_{\text{direct\_move\_to\_p\_cons}}$

[1] In the sense that their effects would work against each other.

[2] Please have a look at Appendix B to see how the composer can personalise the order.

[3] The cost of not using a syncope.

[4] A specific cost for the second species, which applies when a penultimate thesis note does not make a fifth interval with the lowest stratum.

[5] The cost of using sharps or flats.

[6] The cost of having the same note in the downbeat and the upbeat.

[7] The cost of not using a *cambiata* if it is possible. The *cambiata* can be characterised by the following scheme: `consonance - dissonance - consonance`.

[8] The cost of using big or small melodic intervals.

Some notes on the proposed order:

- Two costs come even before the harmonic triad cost: the $\mathcal{C}_{\text{no\_syncope}}$ cost and the $\mathcal{C}_{\text{successive\_p\_cons}}$ cost. Regarding the $\mathcal{C}_{\text{no\_syncope}}$ cost: this cost is at the heart of the fourth species, and a fourth species counterpoint without syncopations is not really a fourth species counterpoint. This is why syncopation is considered even more important than the harmonic triad. And concerning the $\mathcal{C}_{\text{successive\_p\_cons}}$ cost: when Fux expresses his preference for the harmonic triad, he says that there are reasons that are even more important (see rule **1.H8**), and not having successive perfect consonances is one of them.

- The cost of $\mathcal{C}_{\text{penult\_thesis\_is\_fifth}}$ comes before the cost of $\mathcal{C}_{\text{fifths}}$, as it is an exception to the latter (similar to the interaction explained above in the section between $\mathcal{C}_{\text{harmonic\_triad}}$ and $\mathcal{C}_{\text{fifths}}$).

- $\mathcal{C}_{\text{off\_key}}$ was added to its ranking because it is actually an absolute rule not to use off-key notes, but Fux does use some, and so it was decided to put this cost after the very important costs to allow off-key notes to happen.

- The costs $\mathcal{C}_{\text{variety}}$, $\mathcal{C}_{\text{motions}}$ and $\mathcal{C}_{\text{m\_degrees}}$ were ranked in order from least to most restrictive. First we say that we would prefer the note to change as much as possible (with the variety cost), then we indicate our preference for the direction (with the motion cost), and finally we indicate our preference for the size of the motion (with the melodic interval cost). This gives the solver as much flexibility as possible. The other way round would have been more restrictive, since the solver would have minimised the melodic intervals first, setting them all to one, which doesn't leave much room for the motion cost to have an effect, and forcing the variety cost to be high in any case, as with intervals the melody tends to vary only a little.

- $\mathcal{C}_{\text{m2\_eq\_zero}}$ and $\mathcal{C}_{\text{m\_degrees}}$ were classified right after the variety cost as they are an in-measure variation of the variety preference.

NB Please note that using the lexicographic order does not *not* mean that the last costs are not taken into account, they will be *too* minimised by the solver. It just means that if the solver has to choose between minimising one cost or another, it will minimise the first one in the lexicographic order.

4.2.4 Comparison between the three types of costs.

We have now discussed the advantages and disadvantages of each of the three methods. These are all listed in Table 4.1. As you can see, no one method is definitively better than another, and the only way to know which method is better in practice is to *test* them in practice to find out which of the methods gives the best results.

Table 4.1: Comparison of Three Methods According to Criteria

Criteria	Linear Combination	Minimising the maximum	Lexicographic Search
Outstanding costs	Yes	No	Only for minor costs
Sensitivity[9]	No	Some	Yes
One cost might be a bottleneck	No	Yes	No
Need to normalise costs	No	Yes	No
Possibility to ensure a preference of one cost over another	No	No	Yes
Need for hierarchisation of costs	No	No	Yes
Flexibility	Medium	High	Low

Even more, one could think about a combination of all the methods to get rid of their disadvantages. In fact, we could enjoy the advantages of all the methods by combining them and cleverly designing a lexicographic order search in which the cost is a linear combination of a maximum minimisation.

4.3 Experimenting with the three types of costs arrangement

To experiment which method gives the best results, we will follow this plan: first compare the result of a linear combination and the result of a purely lexicographic order, using the default preference order as defined in 4.2.3. We then analyse the result by looking at what could have been done to manage costs more effectively and, where appropriate, group costs together.

These experiments are carried out using different counterpoint combination setups. These setups will increase in complexity, starting with the basic case of 3-voice counterpoint (two counterpoints of the 1st species) and moving on to mixed counterpoints.

The advantage of simple species (first, second and fourth species) is that the search for a solution is much faster. In fact, the search for an optimal solution can be quite time-consuming, and this is even more the case when we are talking about complex species such as the third and the fifth, and when they are combined. This means that it is more difficult to grasp the impact of the cost method when using a setup with complex species. What's more, the vast majority of the costs are related to the interaction of the voices in the first beat of the measures: the behaviour we want to observe, i.e. the interaction between the cost method and the resulting composition, will be just as observable with complex species as with simple ones. Having said that, we are still going to test the cost arrangement methods on all species.

The analyses in this section are superficial and do not deal with in-depth music theory. They will consist of general surface and impression remarks. They are highly subjective and should not be taken at face value. The aim of this analysis is to provide an initial critical view of the results offered by the solver.

The selected *cantus firmi* were chosen from *Gradus ad Parnassum*.

[9] In the sense that changing one cost has a big impact on the result.

4.3.1 Comparing the linear combination and the lexicographic order in practice

First experiment: two first-species counterpoints on two different *cantus firmi*

For this first experiment, if the search time exceeds 30 seconds, the search is stopped and the current solution is analysed.

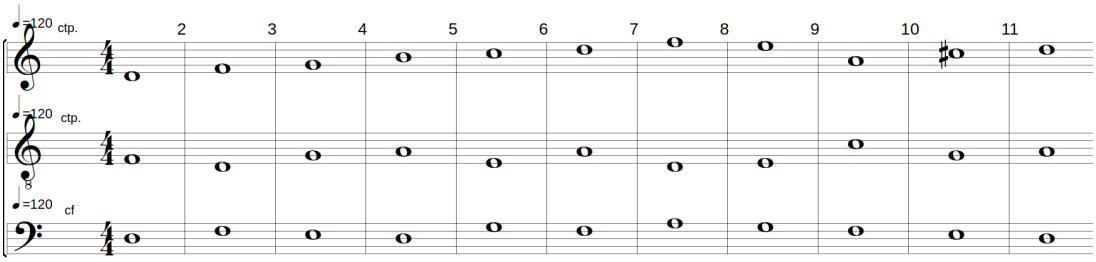

Figure 4.5: Result 1 of the linear combination method with default costs. Click here to listen.

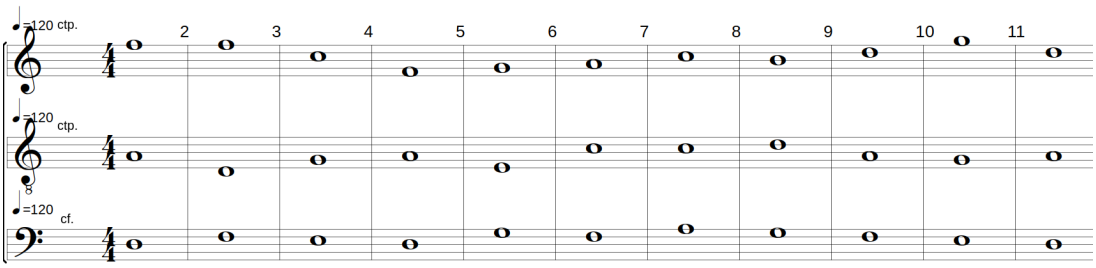

Figure 4.6: Result 1 of the lexicographic search method with default costs. Click here to listen.

Here are the first two results: 4.5 and 4.6. To the average ear, there's not much difference between these two solutions.

In more technical terms, here are the cost arrays:

- for the linear combination: $[2, 20, 0, 3, 4, 6, 4, 14, 0]$, with a total sum of 53,
- for the lexicographical order: $[\underbrace{0}_{\text{succ\_p\_cons}}, \underbrace{14}_{\text{h\_triad}}, 0, 3, 0, 10, \underbrace{16}_{\text{motions}}, 14, \underbrace{8}_{\text{direct\_m\_p\_cons}}]$, with a total sum of 65.

As we can see, the difference between the two is exactly what we expected: high importance costs are prioritised and low importance costs are neglected in the lexicographic search, while the linear combination tries to minimise everything. A good example of this is the harmonic triads: they are more present in the lexicographic solution than in the linear combination (four versus one). Meanwhile, there are seven direct motions and two oblique motions in the lexicographic solution and only two direct motions in the linear combination solution[10].

Let's look at the second result (obtained with a different *cantus firmus*), featured in figures 4.7 and 4.8. Once again, the technical results are fairly as one would expect them:

[10]Please remember that the motions are computed with respect to the lowest sounding note.

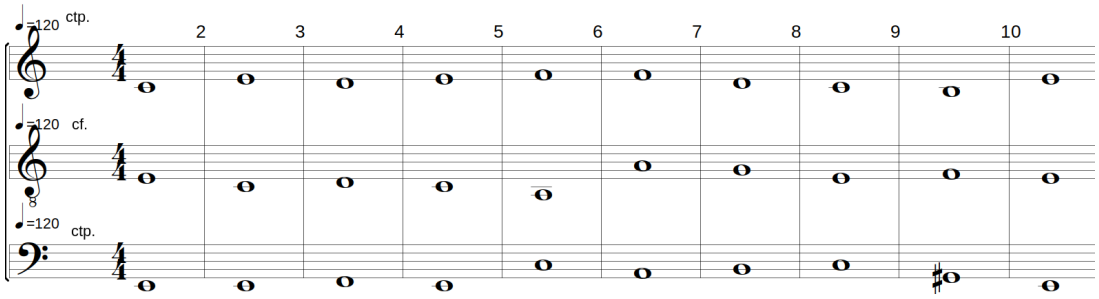

Figure 4.7: Result 2 of the linear combination method with default costs. Click here to listen.

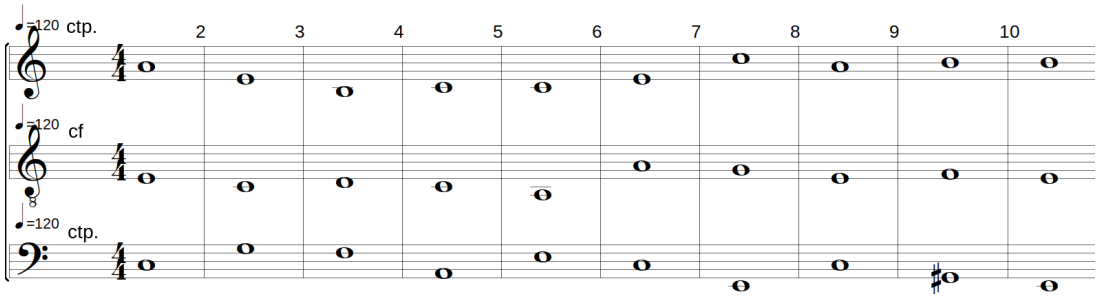

Figure 4.8: Result 2 of the lexicographic search method with default costs. Click here to listen.

- for the linear combination: $[0, 20, 4, 0, 4, 12, 11, 8, 8]$, with a total sum of 67,
- for the lexicographical order: $[0, 16, 0, 2, 4, 8, 10, \underbrace{19}_{\text{melodic\_intervals}}, 8]$, with a total sum of 67.

For the average listener, it is difficult to establish an absolute preference between these two solutions. We will take this same setting and try to mix the techniques for it in section 4.3.2.

Second experiment: one fourth-species counterpoint and one first-species counterpoint on a single *cantus firmus*

If we now go on, here is a result combining the first and the fourth species, and putting the 4th species at the bottom:

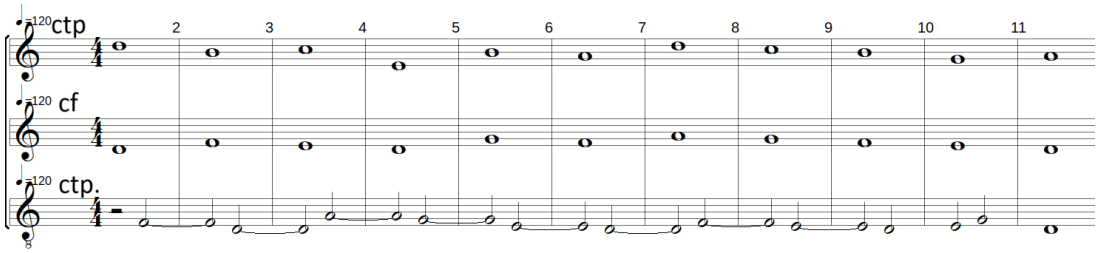

Figure 4.9: Result 3 of the linear combination method with default costs. Click here to listen.

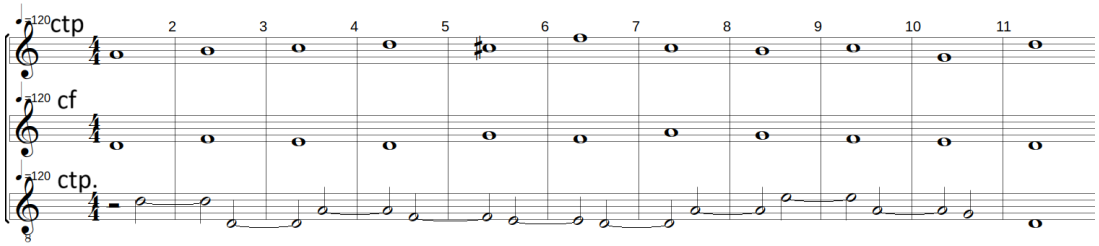

Figure 4.10: Result 3 of the lexicographic search method with default costs. Click here to listen.

Looking at the results obtained with this setup (figures 4.9 and 4.10), we come to the following conclusion: in both cases, the melody is a little dull and lacks dynamism. There is no drastic change in quality between the solutions provided by the two search techniques. However, the solution provided by the lexicographic search is somewhat more exciting, since there is more tension in it and it uses more dissonances and resolvings than the linear combination (even if these resolvings are not the most brilliant).

We immediately notice something else with the 4th species on the bass, which is not related to the costs: there are a few dissonances on the downbeat, as the solver doesn't really take into account the harmonic interaction between the notes of the downbeat of the fourth species and the notes of the downbeat of the other species, but rather the harmonic interaction between the upbeat of the fourth species and the downbeat of the others: which leads to a few surprises, as we can see in these examples (that tension mentioned above).

As far as the costs are concerned, one thing is clear: not all the notes of the 4th species obtained with a linear combination are linked, whereas they all are in the solution of the lexicographic order. This is an obvious consequence of using a linear combination, as this technique is not able to prioritise a cost.

Third experiment: one third-species counterpoint and one second-species counterpoint on a single *cantus firmus*

Our first cross-species test will involve a counterpoint of the 2nd species and a counterpoint of the 3rd species. The *cantus firmus* used is the one proposed by Fux in an example in which he uses exactly these two species. The search was given one minute, as the complexity is getting higher than in the previous experiments.

The results are shown in the figures:

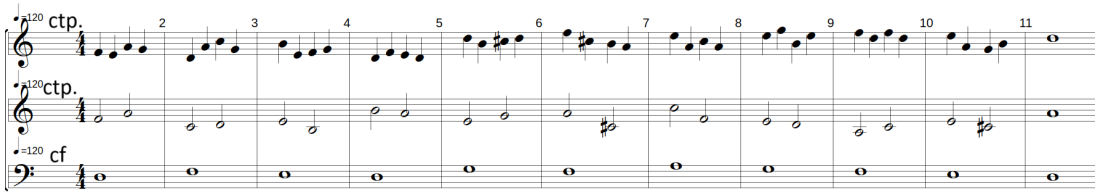

Figure 4.11: Result 4 of the linear combination method with default costs. Click here to listen.

The results are strikingly similar and quite unmelodic. Let's look at these two aspects in turn.

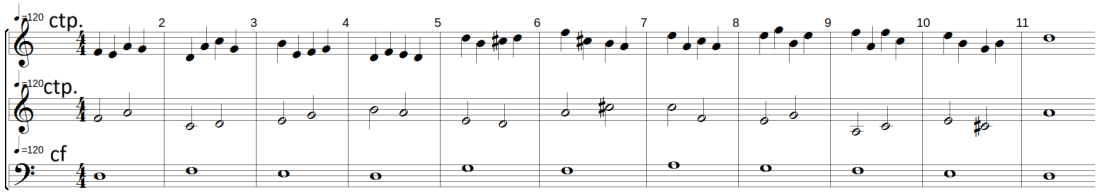

Figure 4.12: Result 4 of the lexicographic search method with default costs. Click here to listen.

About the similarity: The similarity is probably due to two things: the solver doesn't have much room for manoeuvre, since all the voices are highly constrained, so the costs don't have much of an effect in this very setup.

Concerning the lack of melodic quality: it is probably due in part to bad luck (this *cantus firmus* is perhaps particularly difficult to handle) and in part to the lack of constraints linking the upbeats of the various counterpoints. If you think about it, all the rules proposed by Fux in his chapter on three-voice composition link the beats of one voice (*all its beats*) to the first beat of the other voices. This means that there are constraints between the 2nd, 3rd and 4th beats of one voice and the other voices, but never between these 2nd, 3rd and 4th beats of one voice and the 2nd, 3rd and 4th beats of another voice, always with the 1st beat. Obviously, without rules to ensure that the notes of these beats concur[11], it is more complicated for these beats to concur. Of course, it would be wrong to say that it depends only on chance that these beats concur, because that would mean that the beats are independent. Indeed, one might think so at first, because there are no constraints directly linking them, and yet they are linked by their own connections with the first beat of the other voices. In other words, although the third beat of the first counterpoint is not directly linked to the third beat of the second counterpoint, it is indirectly linked to it through the first beat of the second counterpoint: there are constraints between this third beat of the first counterpoint and the first beat of the second counterpoint, and there are also constraints between the first beat of the second counterpoint and the third beat of the second counterpoint. There is therefore a certain dependency and mutual influence between the 3rd beats of the two counterpoints. However, it would be interesting to see if Fux introduces any rules on this subject in his chapter on four-part composition, and if not, it would be interesting to think about what these rules might be in order to maximise the concordance between the notes in the upbeat of the different voices.

Fourth experiment: two fifth-species counterpoints

The fourth experiment is a bit special, as it features two fifth counterpoints *with the same voice range*. This is something Fux doesn't really do in *Gradus ad Parnassum*, but we thought it might be interesting to see how the search method behaves in this situation. Even though Fux doesn't write counterpoints in the same voice range, they are still realistic, for example in the case of a double violin concerto, or any other instrument that has the same voice range and that is played together.

In this experiment, it is interesting to observe how the solver manages the small margin of manoeuvre it has, given that it has to find two counterpoints in the same voice range, all when the counterpoints are forbidden to take the same value (i.e. the same pitch).

[11] The word 'concur' is used here in the same sense that Fux uses it: it means that the notes are somehow put in a relationship that makes them sound good together.

The search was run for two minutes instead of thirty seconds, as the fifth species counterpoint is more complex than the others, and a little more time is needed to let the costs have an effect on the solution. The solutions are those in figures 4.13 and 4.13.

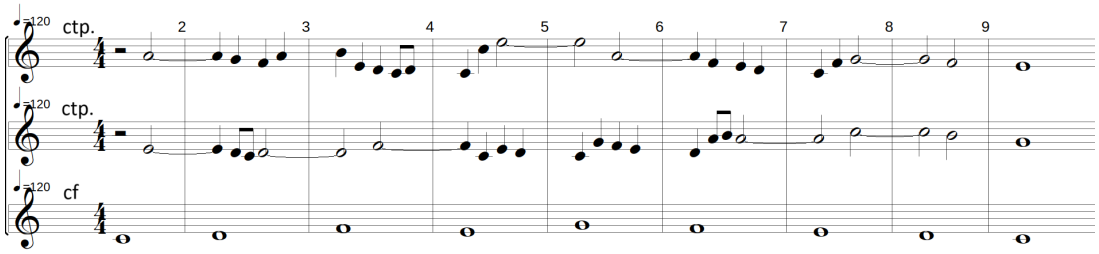

Figure 4.13: Result 5 of the linear combination method with default costs. Click here to listen.

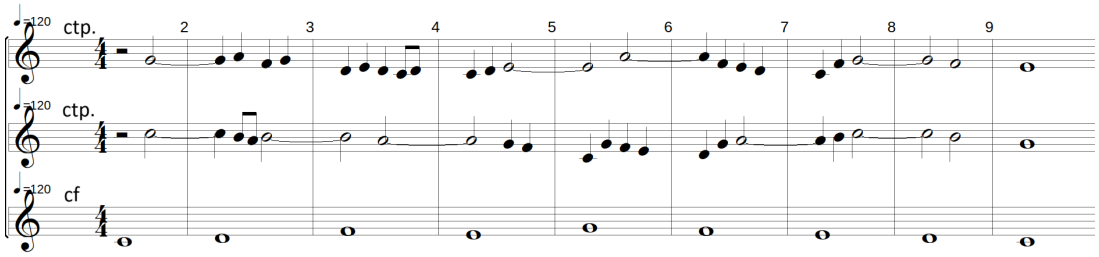

Figure 4.14: Result 5 of the lexicographic search method with default costs. Click here to listen.

Just as for the second experiment, some interesting things happen in the composition found by the lexicographic search. The intervals are more beautiful and the fact that there are dissonances that sometimes resolve gives more meaning to what's going on.

On the other hand, the fact that the penultimate note of the middle voice resolves on a G instead of a C is a bit frustrating, as the final chord feels like it is not a real ending, but this is a recurring problem in all solutions. The reason for this is probably that there is a rule (rule **4.H5**) that makes the solver prefer fifths to octaves, and there is no mention from Fux of deviating from this rule for the last measure.

However, the solver did a surprisingly good job of finding passable solutions with such a small search field. The solutions obtained are far from high art, but you can see the musical intuition behind them.

4.3.2 Mixing the technique of maximum minimisation with lexicographic order

Now what happens when we start to mix the techniques and group some of the costs in the lexicographic order? At each level of the lexicographic order, we calculate the maximum of the costs at that level, which we then try to minimise. All preferences that Fux has not explicitely ordered get packed on the same level, i.e. three levels subsist: the first one, with only the cost for successive perfect consonances, the second one, with only the cost for not using a harmonic triad, and a third one, with all the other costs.

Figures 4.15 and 4.16 show the solutions found by mixing the techniques of lexicographic order and maximum minimisation.

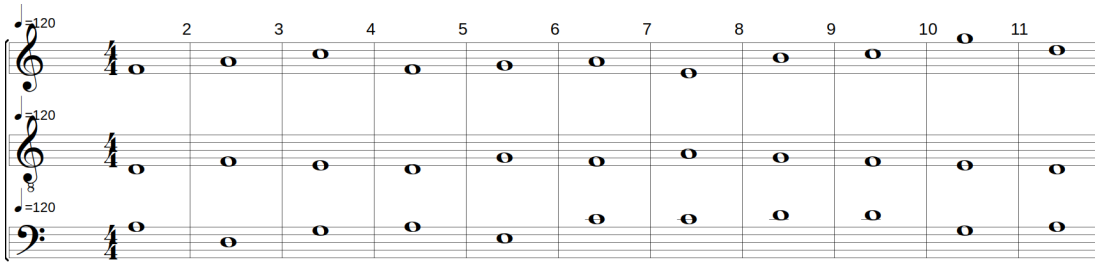

Figure 4.15: Result 1 of a mix between lexicographic and maximum minimisation method. Click here to listen.

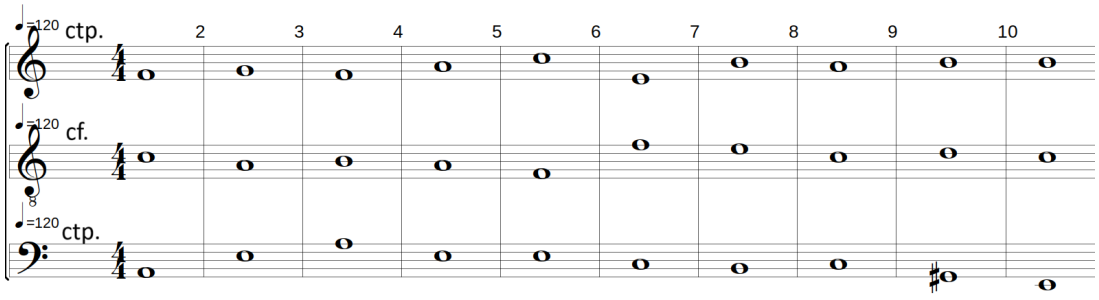

Figure 4.16: Result 2 of a mix between lexicographic and maximum minimisation method. Click here to listen.

Both solutions handle the ending strangely, but this may be a coincidence, as there is no cost that would obviously cause such an ending.

It would be too bold to say that there is a big difference between the results obtained by this method and those obtained by the others. In other words, to the average ear, the solutions provided by this search technique are not significantly different from those provided by the other search methods.

The good news, however, is that the result, far from being excellent, is different. This means that a composer can set up the tool as they wish and get different results from one setup to another. They might start with the default settings and then change one parameter after another until they find what suits them best. This is really good news, because it means that the solution is not too limited to a few possibilities, but that once a valid solution has been found, there is still too much room for personalisation!

Note that the predicted bottleneck effect from section 4.2.2 was indeed observed in both searches, as the solution stopped improving after only ten seconds of search. After these ten seconds, the solver stopped finding better solutions because the third cost level (the one whose maximum was minimised) had already reached its minimum maximum, and so the solver could not distinguish between two solutions if they had the same maximum, since this search technique doesn't allow it. Please refer to figure 4.2 for a better understanding of the situation.

4.4 Conclusion on the search methods

As we have stressed several times in this chapter, there is probably no *best* way of ordering costs. Each technique has its shortcomings, and it is probably by allowing the composer to order their costs as they see fit that the tool will be able to reveal its full potential.

Nevertheless, the lexicographic method seems capable of expressing more charac-

ter than the cost soup of the linear combination method. The intransigent side of the lexicographic method can be adjusted by combining several costs at the same level of the lexicographic order. This combination can be achieved using the method of minimising maxima, but it should be noted that this is only possible to a certain extent if we want to avoid a cost creating a bottleneck on its own. It may also be preferable to combine costs at one level of the lexicographic order by simple addition, at the risk of making certain costs at that level explode. The best compromise that has been found is therefore to give the composer as much choice as possible, by offering them the three possibilities, as well as the combination between the lexicographical order and the other two techniques. This allows the composer to compose their counterpoint iteratively: they make a first attempt that doesn't work well, adjust the costs to direct the search, and so on, until they gets the result that suits them best. In practice, this leads to the user interface described in Appendix B, where a composer can choose their preferred preference order of importance (lexicographic order), then choose if they prefear the costs to be combined in a linear combination or maximum minimisation fashion.

Chapter 5

Musicality of the solutions

This last chapter deals with the musicality of the solutions. While the aim of the previous chapter was to find out how best to adjust the preferences to reflect Fux's will, the aim of this chapter is to see what FuxCP is capable of (using a given preference management technique). We first have a look at basic solutions, i.e. compositions where species are not mixed, as in most of Fux's examples, then we will have a look at what happens when we try to leave Fux's writing style by using preferences, and finally we discuss a bit about cross-species compositions.

5.1 Looking at the results of single species compositions

Composing with three voices, one of which is the *cantus firmus*, one of which is a counterpoint of the first species, and the last of which is a counterpoint of any species, is the basic situation of writing counterpoint, as Fux explains in his teaching of counterpoint. In this first sub-section, therefore, we shall consider five basic counterpoints to a *cantus firmus* in C.

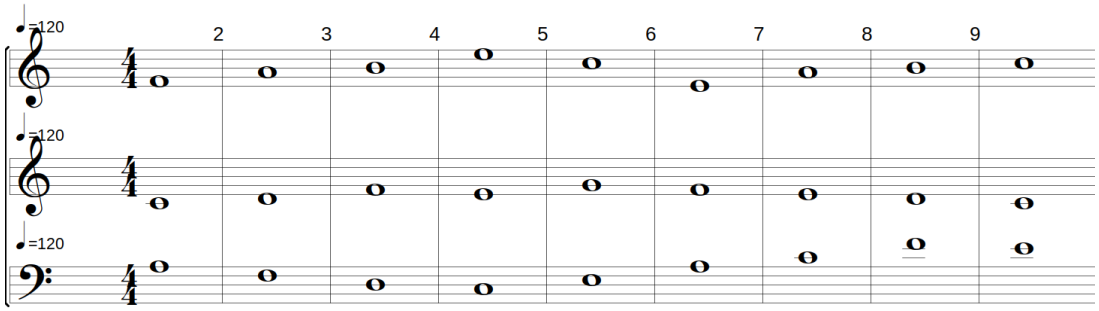

Figure 5.1: Simple first species composition with three voices. Click here to listen.

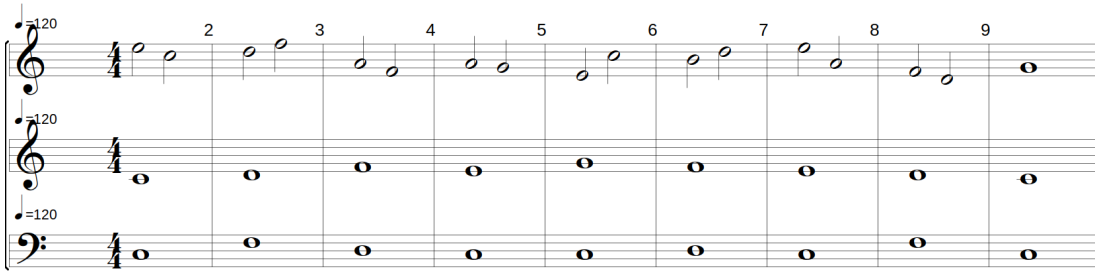

Figure 5.2: Simple second species composition with three voices. Click here to listen.

Looking at all these examples, we can draw the following conclusions:

- Fux was right when he said that three-part composition is much richer than two-part composition (without even having to mix the species). The addition of the

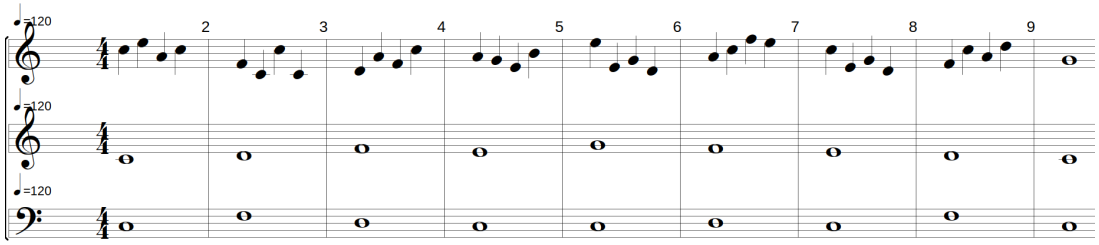

Figure 5.3: Simple third species composition with three voices. Click here to listen.

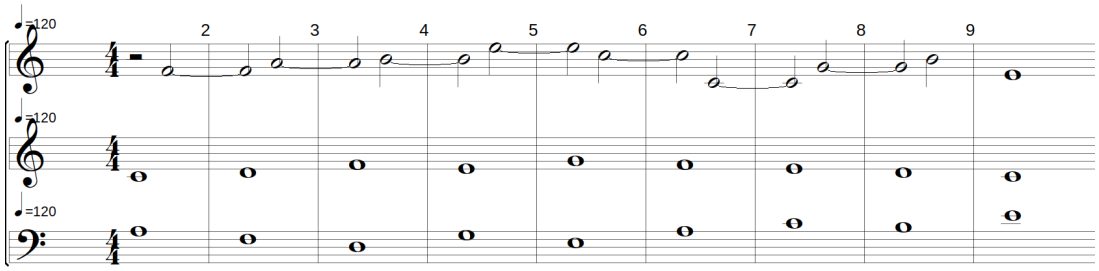

Figure 5.4: Simple fourth species composition with three voices. Click here to listen.

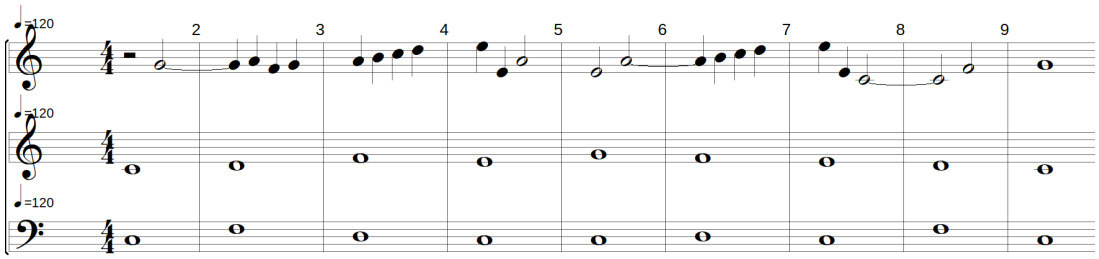

Figure 5.5: Simple fifth species composition with three voices. Click here to listen.

third voice adds depth, dimension and colour, and the problems of monotony that arise in the automation of two-part writing seem to have been partially solved. With two voices, if one voice repeated itself, the composition as a whole suffered from repetitiveness. With three voices, if one voice repeats itself, because the other voices change, this creates variation. In fact, we could say more, as this is in keeping with the general idea of counterpoint, which is that patterns repeat and alternate. In this case, of course, the repetitions are due to chance (remember that all the rules, except the preference for variety, apply to a maximum of one measure, so the solver does not have an overall view of the composition), but in the future we could add rules to encourage these repetitions and variations, or even to encourage the voices to respond to each other!

- Too often the final chord is not conclusive. This could be solved by an additional rule to be defined. It is surprising that Fux does not cover this case in his rules. In fact, the only rules he mentions for the final chord are "no minor third" and "no tenth". Apparently that's not enough.

- It would be difficult to call these compositions masterpieces. However, they stand up well enough to be used as a basis, and with a few personal improvements on the part of the composer, they can be expanded and improved as much as desired. For example, by changing the few notes that don't go together so well,

by modifying the harmonies or by adding different rhythms. The result of the basic counterpoint is therefore more than satisfactory and can be considered a success.

5.2 Trying custom costs

Below is an example of the difference between a counterpoint of the first species, using Fux's preferences (more precisely, using a lexicographic search with default preferences as defined in Section 4.2.3), and a counterpoint in which personal preferences were expressed. These preferences were: to use as few contrary motions as possible, and as many oblique and direct motions as possible. Prioritising this preference (placing it first in the lexicographic order), then prioritising melodic intervals (maintaining a preference for small melodic intervals, as in *Gradus ad Parnassum*), then prioritising variety, and then placing all the other preferences at the penultimate level of the lexicographic order, with the exception of the preference for no successive perfect consonance, which was placed at the very last level.

The search has not been stopped manually; we leave it running until it finds the best solution according to the defined preferences.

The aim of this experiment is twofold: to show that the preferences can have a big impact on the resulting solution, giving the composer a lot of room for manoeuvre in their composition (this was already partially demonstrated in the previous chapter), and to obtain a solution that differs from Fux's compositions. In particular, these small changes in preferences were made to obtain a more monotonous solution, with fewer twists than the Fux-like solutions (for example, for transitions between two parts of a composition, for more quiet moments, ...).

The results of this experiment can be found in figures 5.6 and 5.7.

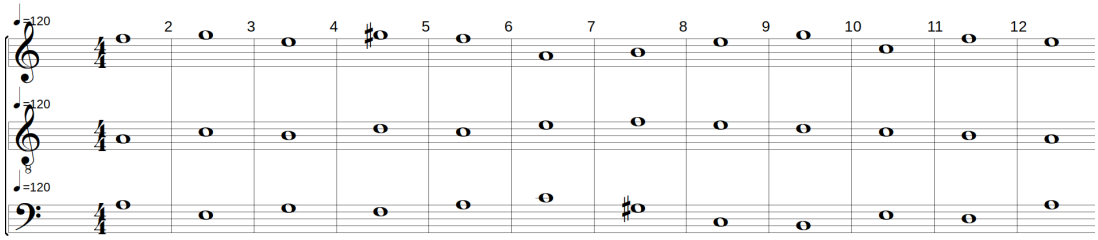

Figure 5.6: Example of a first species counterpoint in three-part composition with Fux's preferences. Click here to listen.

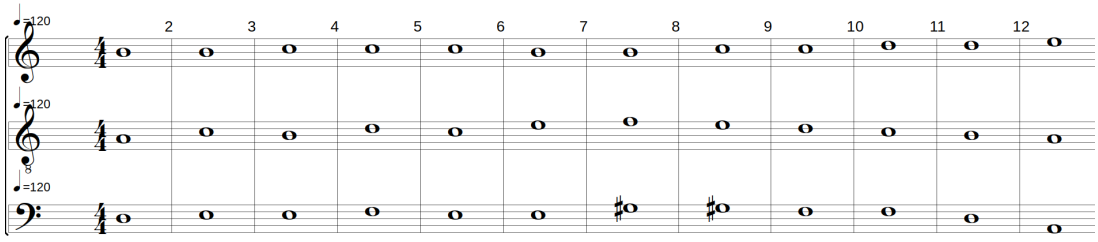

Figure 5.7: Example of a first species counterpoint in three-part composition with custom preferences. Click here to listen.

The result is striking, as the Fux-like solution simply sounds like... Fux, and the custom solution is completely in line with its aim, which was to create a composition

that is more monotonous and where there is less sense of things happening. This example shows something interesting: with the same set of hard constraints (the mandatory rules of Fux), it is possible, thanks to preferences, to make a variety of personal choices that still sound good but deviate from the traditional style of Fux.

5.3 Analysing the results of mixing species

If there's one thing that's clear about multi-species composition, it's that it's a truly unpredictable art. It has already been discussed in the 4.1.3 section that the different interactions between species, voice ranges and *cantus firmi* can take more or less time in terms of solver efficiency. This is also true for the quality of the solutions, and there are times when the solver gives completely correct results, and times when the solution leaves a lot to be desired. This lack of musical quality in the solutions was not the case when we were composing single species counterpoints (without mixing the species), and is something that emerges when combining species. The most plausible hypothesis is that the solver is unable to produce a solution that is *always* beautiful when combining species precisely because Fux has never given any rules specifying how to combine species. We can only hope that such rules exist in the 3rd chapter of his book, or else we'll have to either extrapolate from existing rules or take inspiration from other authors to create these rules.

Below (Figures 5.8 and 5.9) are two examples of counterpoint generated by FuxCP. Both combine counterpoint of the second kind and counterpoint of the fifth species. The search ran for three minutes before being interrupted.

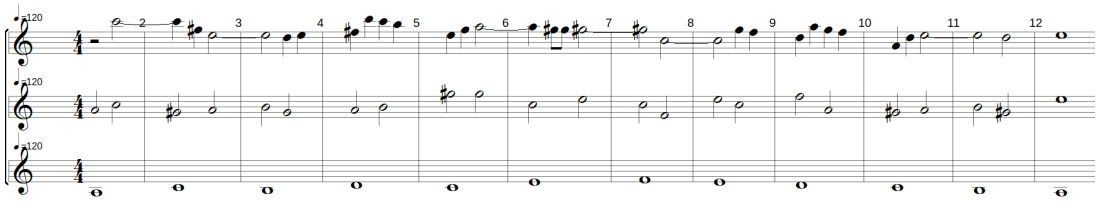

Figure 5.8: Example of a generated counterpoint using a second species counterpoint and a fifth species counterpoint, with in an A scale.

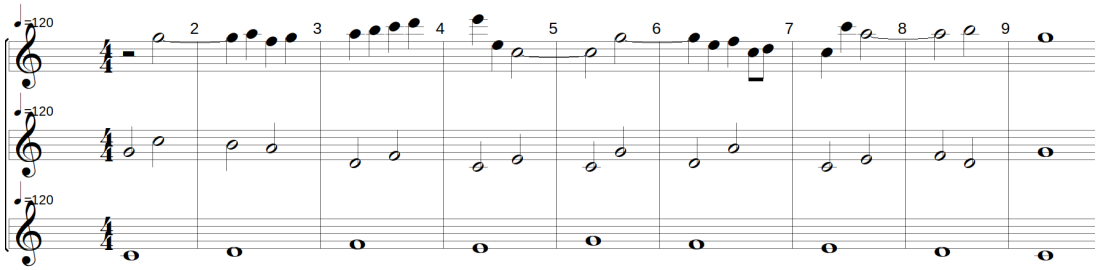

Figure 5.9: Example of a generated counterpoint using a second species counterpoint and a fifth species counterpoint, with in a C scale.

Chapter 6

Conclusion

It is time to look back at the work that has been done, to highlight the progress that has been made, but also the shortcomings and gaps that need to be filled by future improvements. We will therefore use this chapter to discuss some of the key points that emerge from this thesis.

6.1 What has been achieved

...

6.2 Intended use of the FuxCP tool

Throughout the work it is noticeable that all the examples given are quite short (fourteen bars at most). This is largely due to Fux himself, as the examples he gives are all of the same length, presumably for pedagogical purposes. He does not mention this explicitly, but there may also be a practical reason for considering only such short compositions. Indeed, these small compositions can be thought of as 'blocks' which can then be arranged to form a whole. The great advantage of this approach is that the counterpoints between the blocks can be of different kinds, allowing the composition to be constantly renewed.

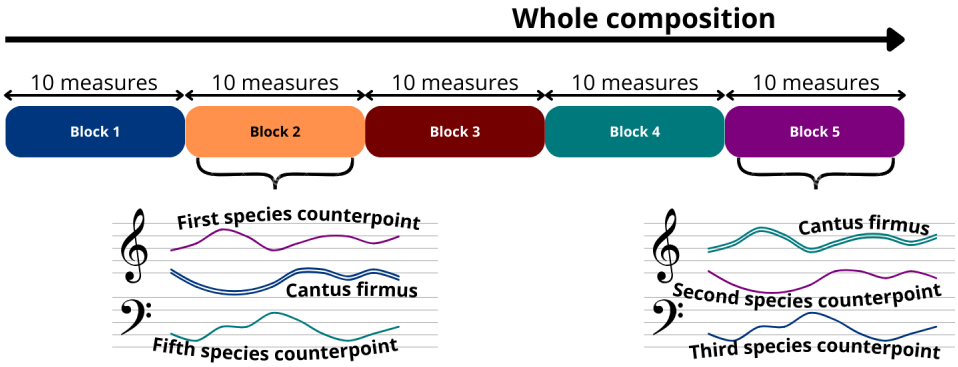

Figure 6.1: Example of what of a composition in blocks could look like

6.3 Known issues about the current state of the work

- As mentioned in section 4.1.3, some few combinations of species, voice ranges and *cantus firmi* cause the solver to fail to find a solution. The current roundabout way to "solve" this is to... change the voice ranges or some other parameter until a structural solution is found. These cases are relatively rare and do not prevent the use of FuxCP.

- If a counterpoint of the fourth species is the lowest stratum, the solver needs more time to find a solution in which all notes are ligated. This is not a problem when combining a fourth species counterpoint with a simple species counterpoint (first or second species), but becomes difficult to handle with more complex species (third or fifth species), as the search time before the solver finds a suitable solution (i.e. with all notes tied) can become very long.

- The current heuristics make the solver branch on the lowest stratum array with a "select the values greater than (min+max)/2" policy. For some reason, it always seems to take the lowest value (or close to it), which means that the melody of the voice that is the lowest stratum is really redundant, something that even variety preference doesn't compensate for on complex species, since the time needed to find a good solution (i.e. one where the variety cost is low, which again means that many notes are different) is too high compared to the time needed to find a passable solution.

  This leads to a second problem, which is that since the downbeat of the part, which is the lowest layer, always consists of the lowest possible note, the following notes (e.g. in the upbeat) will necessarily be higher. This is because the note in the 2nd, 3rd and 4th bars cannot be the same as the note in the 1st bar, but because the 1st bar is already the lowest possible, the notes in the 2nd, 3rd and 4th bars must be higher.

  As this may be difficult to understand, see Figure 6.2, which shows that the second species of counterpoint, in the bass, is composed exclusively of upbeat notes that are higher than the downbeat notes. This is a side effect of the heuristic, and a way should be found to resolve it. Note that in this example the problem is solved very quickly (a matter of seconds) because in this setting the other counterpoint is a first species counterpoint (i.e. the setting is not complex, as explained in 4.1.3) and the and the solver finds solutions where the upbeat is not always higher than the downbeat very quickly.

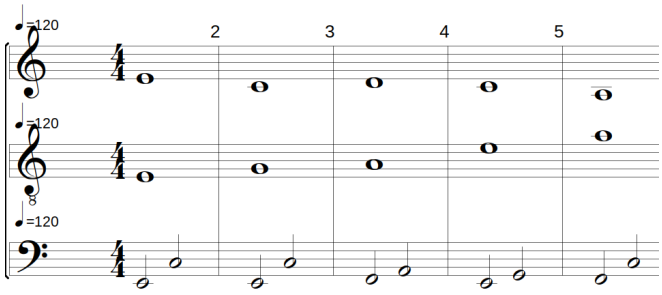

Figure 6.2: Illustration showing the limitation of current heuristics, as the counterpoint of the second species always has a higher upbeat than its downbeat. Note that the composition has been cropped.

6.4 Future work

The work towards fully automated counterpoint composition is progressing, but there is still a long way to go before we can claim to have finished the job. There are several ways to improve the current situation for those who want to improve FuxCP. Here are some ideas for improvement:

1. **Solution quality improvements**

   - *Finalise the formalisation of Gradus ad Parnassum* – The first obvious way to improve the tool is simply to complete the formalisation of Fux's work, which would make it possible to compose in four voices, but also to integrate the additional rules he mentions in his last chapter. This would make it possible to have a complete FuxCP tool, in terms of Fux's rules, and then to supplement Fux's rules with additional rules that could come from any influence. This idea of adding external rules to the Fux rules has already been tested to some extent in the work of T. Wafflard, and the results were more than promising. It is therefore clearly a direction to take in the improvement of the FuxCP tool.
   - *Relate the notes of the 2nd, 3rd and 4th beats to each other* – As mentioned in one of the preference ordering experiments (see 4.3.1), there are no direct constraints between the 2nd, 3rd and 4th beats of each counterpoint. The only way they influence each other is through the transitivity of the constraints: A and B are constrained together, and so are B and C, so A and C are connected in some way. The reason there are no direct constraints is that Fux didn't mention any. Anyway, this leads to some unmelodicity, and it is definitely a good idea to find some rules (e.g. from other authors) that could deal with this unmelodicity.
   - *Address any of the current limitations* – The section 6.3 discusses some limitations and issues with the current state of the work. Solving any of them would be an improvement for FuxCP.

2. **Software architecture**

   - *Migrate the project to C++* – Gecode is written in C++, and C++ is a language much better suited to managing implementations like FuxCP. GiL works really well, but has shown its limitations more than once: way too verbose, hard to manage objects (which are useful for designing FuxCP) since using Lisp, and lacking some of Gecode's features. These reasons alone are a huge incentive to migrate the whole implementation to C++. This would make it possible to further improve the implementation with more convenience and efficiency.
     Here we repeat the words of T. Wafflard, who had already reached the same conclusion:

     > "Currently, constraints are added to a species via a long function that dispatches the constraints, rather than via class inheritance. Ideally, object-oriented inheritance should be used to represent the different variable arrays and species. All variable arrays (H, M , P , etc.) have something in common, whether in terms of their size rel- ative to the cantus firmus, or in terms of the way certain rules are applied. A relatively abstract class should represent this type of array to enable these commonalities to be brought together.

> The same applies to species that share common rules and should have been represented in a class system of their own. It would be logical for species to be children of the first species. Unfortunately, the scope of this work does not allow for a complete overhaul of the architecture. Moreover, in the near future, the entire code may have to be redone in C++ for reasons of performance, features, maintainability, and so on. Also, GiL has reached its limits, both in terms of ease of programming and in terms of possibilities. The Lisp language is not designed for writing mathematics, since each operation requires a different function call. Code readability can become complicated because these calls are all represented by parentheses. At the same time, it is not possible with GiL to combine basic mathematical operations to form a larger one. One has to break down each complex operation into simple intermediate basic operations a bit like writing assembly, which is undesirable for larger projects. Not to mention that branch-and-bound, heuristics, and multithreading seem complicated to implement in GiL." [1, p.67]

3. **Solver performance**

   - *Bettering the heuristics and reorganising the constraints* – Obviously, increasing the speed at which the solver finds solutions increases the speed at which the solver finds good solutions. It is therefore crucial to continue working on the heuristics to find better and better solutions. At the same time, once the globality of *Gradus ad Parnassum* has been formalised, it might be interesting to rethink the constraints that apply to the composition in an intelligent way, to make the solver's work easier and to have a set of constraints that hold together better.

Bibliography

[1] Thibault Wafflard. "FuxCP: a constraint programming based tool formalizing Fux's musical theory of counterpoint". Prom. by Peter Van Roy. MA thesis. École polytechnique de Louvain, Université catholique de Louvain, 2023. URL: http://hdl.handle.net/2078.1/thesis:40739.

[2] Damien Sprockeels et al. "A Constraint Formalization of Fux's Counterpoint". In: *Journées d'Informatique Musicale (JIM)* (2023).

[3] Klaus-Jürgen Sachs and Carl Dahlhaus. *Counterpoint*. Oxford University Press. 2001. DOI: 10.1093/gmo/9781561592630.article.06690. URL: https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000006690.

[4] Juliet Hess. "Balancing the counterpoint: Exploring musical contexts and relations". In: *Action, Criticism & Theory for Music Education* 15.2 (2016), p. 50.

[5] Richard Kramer. "Gradus ad Parnassum: Beethoven, Schubert, and the romance of counterpoint". In: *19th-Century Music* 11.2 (1987), pp. 107–120.

[6] Jaime Altozano. *De Pokemon a Bach. Una historia de voces.* Language: Spanish. July 2017. URL: https://youtu.be/Mr8ICnGutYM?si=6W4XpNx-lTgJtaap.

[7] Heinrich Schenker. *Kontrapunkt*. German. Vol. 2: iss.1. Vol. 2 of Neue musikalische Theorien und Phantasien. Stuttgart: J.G. Cotta'sche Buchhandlung Nachfolger, 1906.

[8] Knud Jeppesen and Glen Haydon. *Counterpoint: The Polyphonic Vocal Style of the Sixteenth Century*. English. Englewood Cliffs, N. J.: Prentice-Hall, 1960.

[9] David Gaynor Yearsley. *Bach and the Meanings of Counterpoint*. English. Cambridge ; New York: Cambridge University Press, 2002.

[10] Richard L. Crocker. "Discant, Counterpoint, and Harmony". In: *Journal of the American Musicological Society* 15.1 (1962), pp. 1–21. ISSN: 00030139, 15473848. URL: http://www.jstor.org/stable/830051.

[11] Bill Schottstaedt. *Automatic Species Counterpoint*. Research Report STAN-M-19. Departement of Music, Stanford University, 1984. URL: https://ccrma.stanford.edu/files/papers/stanm19.pdf.

[12] John Polito, Jason M Daida, and Tommaso F Bersano-Begey. "Musica ex machina: Composing 16th-century counterpoint with genetic programming and symbiosis". In: *Evolutionary Programming VI: 6th International Conference, EP97 Indianapolis, Indiana, USA, April 13–16, 1997 Proceedings 6*. Springer. 1997, pp. 113–123.

[13] Andres Garay Acevedo. "Fugue composition with counterpoint melody generation using genetic algorithms". In: *International symposium on computer music modeling and retrieval*. Springer. 2004, pp. 96–106.

[14] Gabriel Aguilera et al. "Automated generation of contrapuntal musical compositions using probabilistic logic in Derive". In: *Mathematics and Computers in Simulation* 80.6 (2010), pp. 1200–1211. ISSN: 0378-4754. DOI: https://doi.org/10.1016/j.matcom.2009.04.012.

[15] Dorien Herremans and Kenneth Sörensen. "Composing first species counterpoint with a variable neighbourhood search algorithm". In: *Journal of Mathematics and the Arts* 6.4 (2012), pp. 169–189. DOI: 10.1080/17513472.2012.738554.

[16] Maciej Komosinski and Piotr Szachewicz. "Automatic species counterpoint composition by means of the dominance relation". In: *Journal of Mathematics and Music* 9.1 (2015), pp. 75–94. DOI: https://doi.org/10.1080/17459737.2014.935816.

[17] Ars Nova Software. *Counterpointer*. Developed and produced by Ars Nova Software. Address: PO Box 3333, Kirkland, WA 98083-3333, USA. Available on Microsoft Store and Mac App Store. 2019. URL: https://www.ars-nova.com/counterpointer3.html.

[18] Alex Ding et al. *Counterpointer*. Project for Brown University's CSCI 0320 course. 2021. URL: https://github.com/counter-pointer/counterpointer.

[19] Baptiste Lapière. "Computer-aided musical composition Constraint programming and music". Prom. by Peter Van Roy. MA thesis. École polytechnique de Louvain, Université catholique de Louvain, 2020.

[20] Damien Sprockeels. "Melodizer: A Constraint Programming Tool For Computer-aided Musical Composition". Prom. by Peter Van Roy. MA thesis. École polytechnique de Louvain, Université catholique de Louvain, 2021.

[21] Clément Chardon, Amaury Diels, and Federico Gobbi. "Melodizer 2.0: A Constraint Programming Tool For Computer-aided Musical Composition". Prom. by Peter Van Roy. MA thesis. École polytechnique de Louvain, Université catholique de Louvain, 2022.

[22] Alfred Mann. *The Study of Counterpoint. From Johann Joseph Fux's Gradus ad Parnassum*. English. Ed. and trans. Latin by Alfred Mann. Revised Edition. 500 Fifth Avenue, New York, N.Y. 10110: W.W. Norton & Company, 1971. ISBN: 0-393-00277-2. URL: http://www.opus28.co.uk/Fux_Gradus.pdf.

[23] Simonne Chevalier. *Gradus ad Parnassum. Johann Joseph Fux*. French. Ed. by Gabriel Foucou. Trans. Latin by Simmone Chevalier. 2019. ISBN: 978-2-9556093-6-1.

[24] Johann Joseph Fux. *Gradus ad Parnassum*. French. Trans. by Pierre Denis. Paris: Diod, Bijoutier, Garnier & Cadet, 1773. URL: https://s9.imslp.org/files/imglnks/usimg/b/b2/IMSLP231222-PMLP187246-fux_traite_de_composition_1773.pdf.

[25] Johann Joseph Fux. *Gradus ad Parnassum*. German. Ed. and trans. by Lorenz Christoph Mizler. Leipzig: Mizler, 1742. URL: https://s9.imslp.org/files/imglnks/usimg/6/67/IMSLP273867-PMLP187246-gradusadparnassu00fuxj.pdf.

[26] Johann Joseph Fux. *Gradus ad Parnassum*. Reprinted in 1966 by Broude Bros., New York. Vienna: Johann Peter van Ghelen, 1725. URL: https://s9.imslp.org/files/imglnks/usimg/f/fd/IMSLP91138-PMLP187246-Fux_-_Gradus_ad_Parnassum.pdf.

[27] Francesca Rossi, Peter Van Beek, and Toby Walsh. "Constraint programming". In: *Foundations of Artificial Intelligence* 3 (2008), pp. 181–211.

[28] Roman Barták. *Constraint Programming*. First. First Edition. Creative Commons License, 1998.

[29] David R Morrison et al. "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning". In: *Discrete Optimization* 19 (2016), pp. 79–102.

[30] IRCAM STMS Lab. *OpenMusic*. Version 7.2. URL: https://openmusic-project.github.io.

[31] Guido Tack and Mikael Zayenz Lagerkvist. *Generic Constraint Development Environment*. Version 6.2.0. Gecode. URL: https://www.gecode.org/index.html.

[32] James Bielman and Luís Oliveira. *CFFI: The Common Foreign Function Interface*. 2005. URL: https://cffi.common-lisp.dev/.

[33] B. Mcnair. *Understanding Stratum in Geology, Including Types and Examples*. Accessed on December 3, 2023. 2023. URL: https://geologybase.com/stratum/.

[34] Noël Gallon and Marcel Bitsch. *Traité De Contrepoint*. French. Ed. by Durand et Cie. Paris, 1964. URL: https://artinfuser.com/exercise/md/pdf/Gallon-Bitsch-Contrepointe.pdf.

[35] Walter Piston. *Harmony: Fifth Edition*. English. Ed. by Mark DeVoto. 5th. W. W. Norton & Company, 1987, p. 32.

Appendix A

Software Architecture

This appendix summarises the architecture of the software. The first figure (A.1) shows how FuxCP is integrated into the tools it uses, and the second figure (A.2) shows the internal structure of FuxCP.

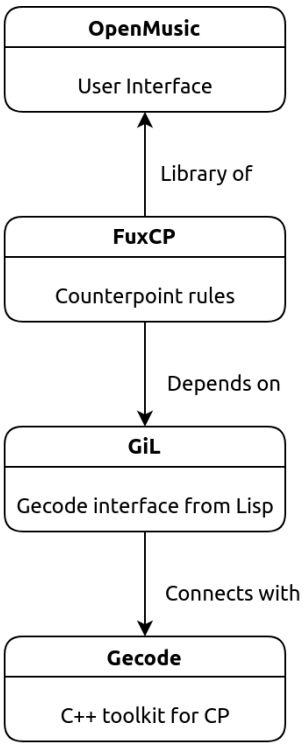

Figure A.1: Integration of FuxCP within the other tools

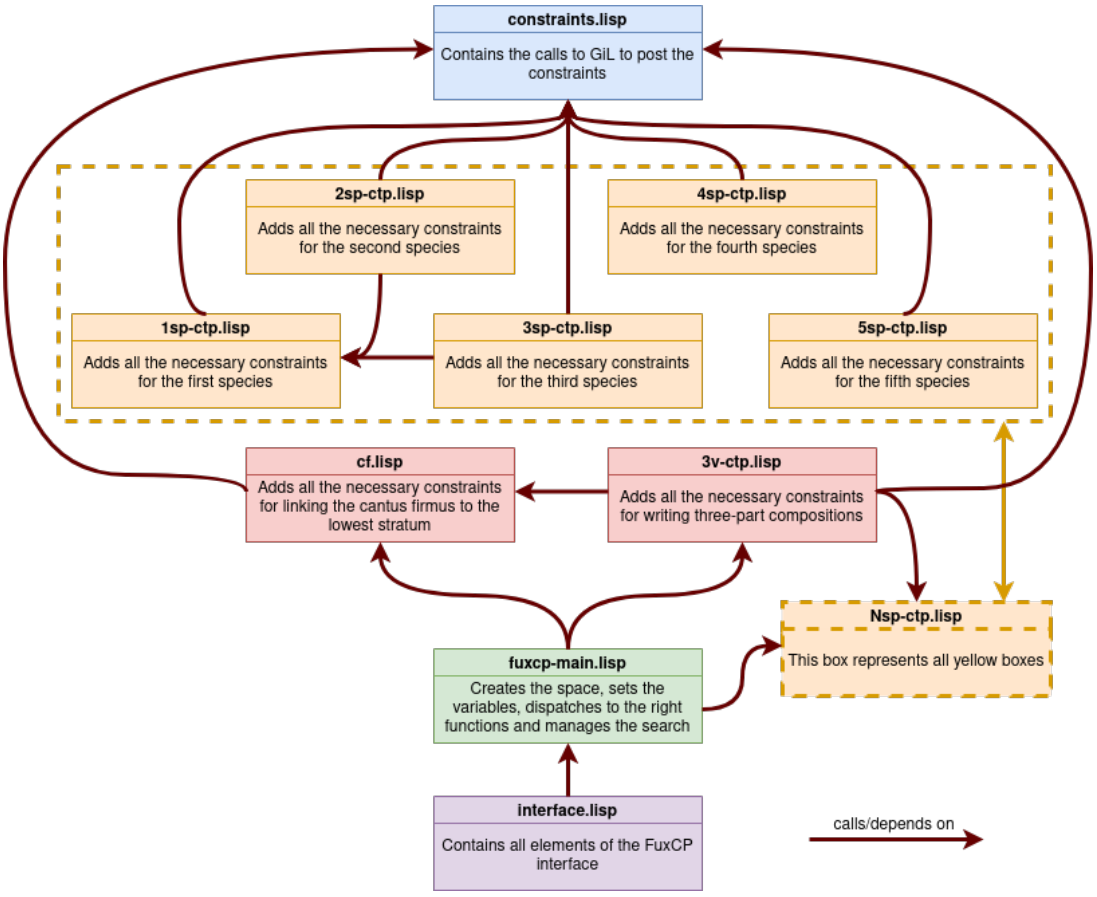

Figure A.2: Internal structure of FuxCP

Appendix B

User Guide

This manual provides an overview of FuxCP, covering the installation process, its use within OpenMusic and a description of the costs displayed in the interface. Although FuxCP is designed to be compatible with all platforms, it relies on GiL, which currently only works on MacOS and Linux. Unfortunately, GiL does not support Windows due to compatibility issues between the 32-bit Lisp licence used by OpenMusic and the 64-bit Gecode Windows version. Although it is technically possible to obtain a 32-bit version of Gecode for Windows, this is not recommended.

B.1 Installing FuxCP

B.1.1 Prerequisites

To use FuxCP you need to download and install the following tools:

- Gecode : `https://www.gecode.org/download.html/`
- OpenMusic : `https://openmusic-project.github.io/openmusic/`

And download the following libraries:

- GiL : `https://github.com/sprockeelsd/GiLv2.0/`
- FuxCP : `https://github.com/sprockeelsd/Melodizer/`

There are other tools available on the latest GitHub, such as Melodizer and Melodizer2.0. For the purposes of this guide, only the FuxCP folder is needed.

B.1.2 Loading FuxCP in OpenMusic

In order to use the above libraries, OpenMusic must be running. When opening any workspace, locate the toolbar at the top of the interface. Click on the "Windows" button, highlighted in the figure B.1, and select "Library" from the drop-down menu. This will bring up a new window. From the toolbar of this window, select 'File' and then 'Add Remote Library'. Navigate through your file system to find the path where the previously downloaded FuxCP and GiL libraries are stored. Once located, the libraries should appear under the "Libraries" folder in the "Library" window, as shown in Figure B.2. Right click on "fuxcp" and select "Load Library". If no errors occur, the setup is complete.

However, if an error occurs, it may be a linking problem with the Gecode library. For MacOS users, a script from the `c++` folder of the GiL library can be used. Edit the path to Gecode within the script to match your system configuration. Linux users should add the Gecode library to the `LD_LIBRARY_PATH` variable. Go to the `/etc/ld.so.conf.d` folder and create a new `.conf` file if one does not already exist. In this file, add the full path to the Gecode library, save it, and run `sudo ldconfig` to update the system with the new library. Don't forget to restart OpenMusic and don't lose hope. Following these steps should ensure that FuxCP works properly.

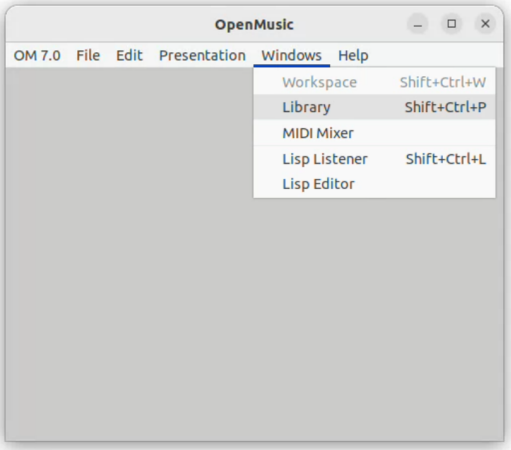

Figure B.1: Opening the "Library" window in OpenMusic.

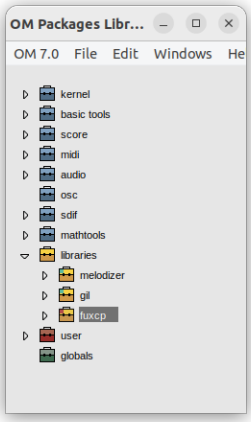

Figure B.2: Loading the "fuxcp" library in OpenMusic.

B.2 Using FuxCP in OpenMusic

Setup

Using FuxCP in OpenMusic is straightforward. There is a single block that contains the entire graphical interface of the tool. This block or class is called `cp-params`. To load it, right click anywhere in the patch and select $Classes \rightarrow Libraries \rightarrow FuxCP \rightarrow Solver \rightarrow CP - PARAMS$. Alternatively, you can just double-click anywhere in the patch, type "fuxcp::cp-params" and press Enter. This also works for "poly", "voice" and "x-append".

Once this block has appeared, all you have to do is bind an OM voice object, representing the *cantus firmus*, to the second argument of `cp-params` as shown in figure B.3. Don't forget to block the input voice object and evaluate `cp-params` so it can detect the new input. Now `cp-params` can be blocked too. From now on, you could directly use the interface and generate counterpoints using the tool. If you want to retrieve the voice object containing the counterpoint generated by the tool, just bind the third argument on the output side to a voice object. Once bound, it is then possible to evaluate the voice object so that it updates.

If you want to get the whole composition in one object, you have to do some fiddling with OpenMusic. The simplest way to do this is shown in Figure B.3, and works as follows: get the POLY object returned by CP-PARAMS on its third output, split this object in two (the two voices), then get the *cantus firmus* , which is the second output of CP-PARAMS, and put all the voices back together in the desired order using x-append functions.

Listening to the solution

OpenMusic does not have built-in sound. You will need to use a third party application to listen to the result. Here is a solution that works to listen to the music: having installed TiMidity++[1], run the following command before opening OpenMusic:

```
timidity -iA -B2,8 -Os
```

Then go to *OpenMusic* → *Preferences* → *MIDI* → *Ports setup* → *Output devices* and select "TiMidity port 0".

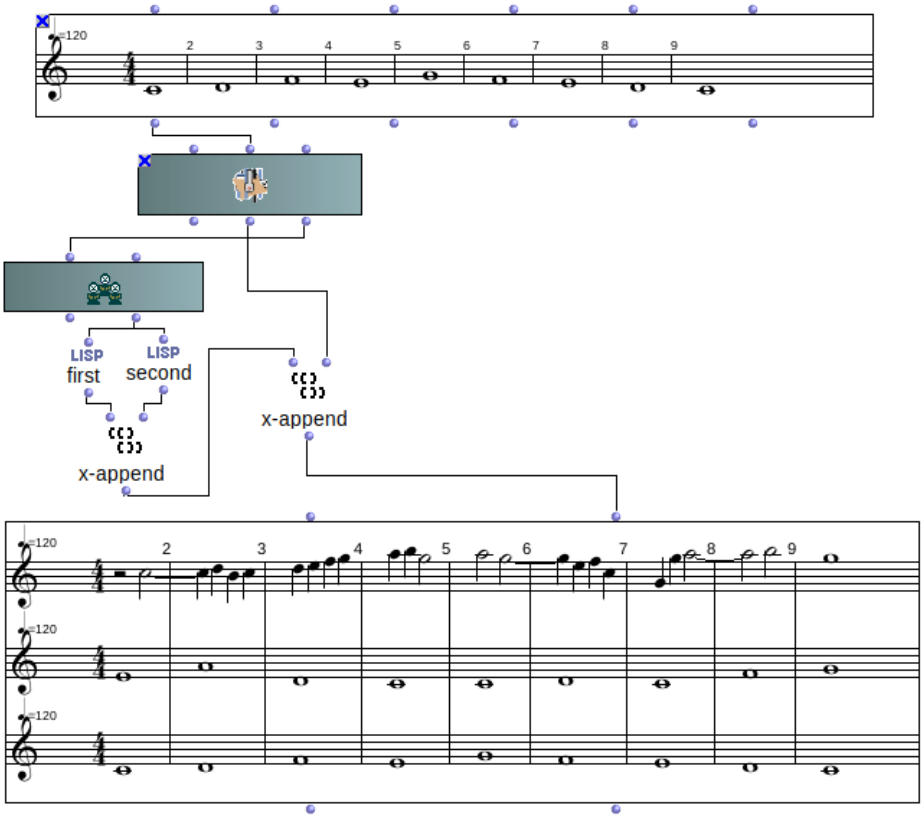

Figure B.3: View of a patch using `fuxcp::cp-params` in OpenMusic.

Counterpoint generation

But how do you use the interface? Simply double-click on the block to bring it up. The interface is sorted from left to right, so the preferences are divided into three different categories: "General preferences", "Melodic preferences", "Species-Specific Preferences", "Solver Configuration" and, in the bottom right corner, "Solver Launcher" (see figure B.4). Once you have chosen the preferences, the default ones representing the

[1]TiMidity++ is a software synthesizer that can play MIDI files without a hardware synthesizer, click here to install.

Fux style, you have to save the parameters ("Save Config") in order to start the search for a solution ("Next Solution"). This search may take a fraction of a second, or it may take tens of minutes if the parameters chosen make the search difficult. If the search takes too long, you can always stop it by clicking on "Stop". You can then either change the settings in some way (often by changing the voice range).

You will notice that there are almost always two settings for each preference. The first is the importance: it corresponds to the priority the solver will give to reducing the value of this cost. An importance of 1 means that it will be the absolute priority of the solver, whereas a preference of 14 means that the solver will minimise this cost if it doesn't affect the other costs. The second setting is the value: it determines the actual value of the cost corresponding to the preference. It is very useful when two preferences are set to have the same importance, in which case their respective cost values will have an effect. For example, if cost A and cost B both have an importance of 1, but cost A has a very high cost compared to B, A will affect the quality of the solution more than B, even though they both have the same importance. If two costs have the same importance, the yellow panel (bottom left) allows you to choose how to combine them: either by linear combination or by maximum minimisation. For more information, see Chapter 4 of this thesis that explains how costs work in detail.

All costs are explicitly defined in the following section, in Table B.1.

Pressing the "Next Solution" button will display the solution as a pop-up. What appears on the screen are the two counterpoints. The *cantus firmus* must be added manually as described at the beginning of this section.

The other option is to press the "Best Solution" button. This will start an infinite search that will only stop when the best solution has been found (which can take hours). You can evaluate the output object at any time to see what the best result is so far, and this will not stop the search, so you can see how the solution improves step by step, and stop the search when it has produced something you are happy with.

Please note that the preferences do not affect the speed of finding the first solution. The first solution is the first valid solution and is not affected by the costs. Only the subsequent solutions can be affected by the preferences.

B.3 Interface Parameters Description

Table B.1 describes all the parameters available in the interface. A low cost represents a high preference while a high cost represents a low preference.

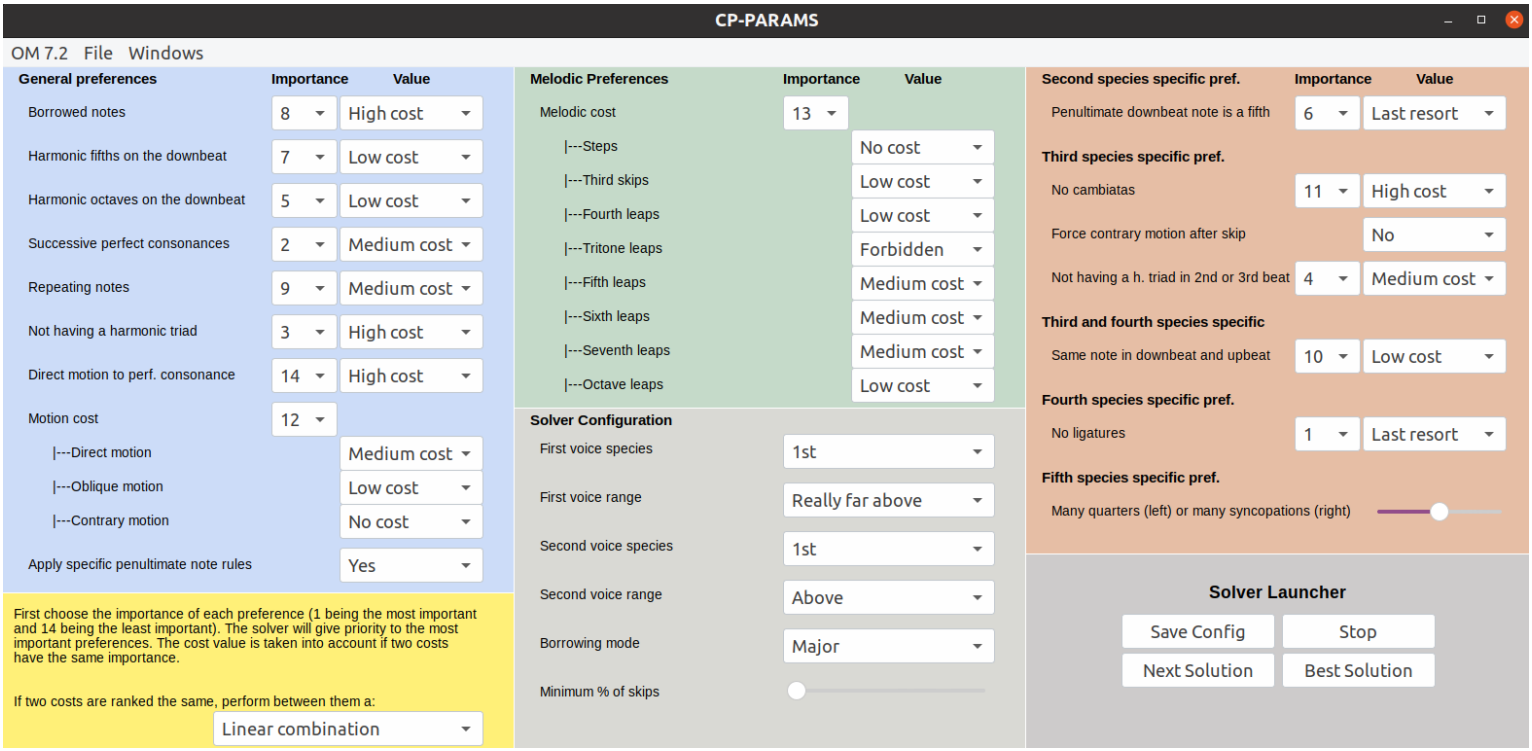

Figure B.4: User interface of the `fuxcp::cp-params` class in OpenMusic.

Name	Description	Default value
Borrowed notes	Preference for borrowed notes outside the diatonic scale. A high cost means as few borrowed notes as possible.	High cost
Harmonic fifths on the downbeat	High cost means as few harmonic fifths on the downbeats as possible.	Low cost
Harmonic octaves on the downbeat	High cost means as few harmonic octaves on the downbeats as possible.	Low cost
Successive perfect consonances	High cost means as few successive perfect consonances as possible.	Medium cost
Repeating notes	High cost means as few repeating notes as possible, i.e. as many different notes as possible. This cost corresponds to the variety cost.	Medium cost
Not having a harmonic triad	High cost means as many harmonic triads as possible.	Medium cost
Direct motion to perf. consonance	High cost means as few direct motions to perfect consonances as possible.	Last resort
Direct motion	High cost means as few direct motions as possible.	Medium cost
Oblique motion	High cost means as few oblique motions as possible.	Low cost
Contrary motion	High cost means as few contrary motions as possible.	No cost
Apply specific penultimate note rules	Force all rules on the notes of the penultimate measure. This applies only to two-part composition and refers to the penult. note having to be a major sixth or a minor third.	Yes
Steps	High cost means as few steps as possible.	No cost
Third skips	High cost means as few third skips as possible.	Low cost
Fourth leaps	High cost means as few fourth leaps as possible.	Low cost
Tritone leaps	High cost means as few tritone leaps as possible.	Forbidden
Fifth leaps	High cost means as few fifth leaps as possible.	Medium cost
Sixth leaps	High cost means as few sixth leaps as possible.	Medium cost
Seventh leaps	High cost means as few seventh leaps as possible.	Medium cost
Octave leaps	High cost means as few octave leaps as possible.	Low cost
2nd: Penult. downbeat note is a fifth	High cost means trying to ensure that the penultimate downbeat is not a fifth.	Last resort
3rd: No cambiatas	A high cost means as many cambiatas as possible	High cost
3rd: Force contrary motion after skip	Force that a melodic skip or leap is followed by a melodic step in a contrary motion.	No
3rd& 4th: Same note in downbeat and upbeat two beats apart	High cost means as many differrent notes in the downbeat and upbeat.	Low cost
4th: No ligatures	High cost means as few not-ligatured notes, i.e. as many ligatures as possibles.	High cost
5th: Many quarters or many syncopations	Determines the minimum percentage of quarter notes or syncopations in the fifth species. Pushing the slider all the way to one side is not recommended.	
Voice species	Determines the type of counterpoint that the tool will generate.	1st and 1st
Voice range	Determines around which pitch the counterpoint will be generated depending on the pitch of the first note of the *cantus firmus*.	Above and very far above
Minimum % of skips	Determines, depending on the counterpoint size, the percentage of melodic intervals larger than one step.	0%
Borrowing mode	Type of scale from which notes can be borrowed to generate counterpoint. The first note of the *cantus firmus* determines the tonic of this scale. Applies everywhere except the penultimate bar.	Major
Save Config	Saves all established preferences and allows you to start a new search for this configuration later.	-
Next Solution	Starts or continues the search for the previously saved configuration. Displays a new window with the first better solution found. Displays an error message if no solution can be found.	-
Stop	Pause the search. This may take up to 5 seconds to take effect.	-
Best Solution	Starts or continues the search for the previously saved configuration. Does not display a window, but returns the best solution found so far, accessible by evaluating the output of `cp-params`. Displays an error message if no other solution can be found.	-
Linear combination or maximum minimisation	Choose whether the equally important costs are combined according to a linear combination or according to a maximum minimisation. More details about it in Chapter 4.	Linear combination

Table B.1: Description of the parameters of `fuxcp::cp-params`.

Appendix C

Complete set of rules for two and three part compositions

This appendix contains all the constraints for composing counterpoint for two or three voices. This appendix contains only the formalised equations, the corresponding explanations can be found in the corresponding sections of this thesis and that of T. Wafflard.

All the rules apply to three-voice compositions, but only the rules for two voices apply to two-voice compositions. Rules for three-part compositions are indicated by '3V' at the beginning of the rule.

Some of the rules are different depending on whether the composition is for two or three voices. In these cases, the mathematical relationship to be followed for the rule in question, depending on the situation, is clearly indicated.

Implicit General Rules of Counterpoint

G1 *Harmonic intervals are always calculated from the lower note.*

Already handled by making the difference value absolute for the **H** variable.

G2 *The number of measures of the counterpoint must be the same as the number of measures of the cantus firmus.*

Listing C.1: Definition of N in the first species.

```
        (defvar *notes (list nil nil nil nil))
        ; ...
        ;; FIRST SPECIES ;;
        ; setting the first list of *notes with
        ;   integer *cf-len as size
        ;   set *extended-cp-domain as available notes
        (setf (first *notes)
            (gil::add-int-var-array-dom *sp* *cf-len *extended-cp-domain))
```

G3 *The counterpoint must have the same time signature and the same tempo as the cantus firmus.*

Listing C.2: Definition of N in the first species.

```
    (defvar *notes (list nil nil nil nil))
    ; ...
    ;; FIRST SPECIES ;;
    ; setting the first list of *notes with
    ;   integer *cf-len as size
    ;   set *extended-cp-domain as available notes
    (setf (first *notes)
        (gil::add-int-var-array-dom *sp* *cf-len *extended-cp-domain))
```

G4 *The counterpoint must be in the same key as the cantus firmus.* This rule is already handled by the creation of the set $\mathcal{N}$. The example of the actual rule given above will clarify the explanations. Let k be the value of the key determined by the key signature, i.e. 60 for C; and t the tonic of the piece, i.e. $Cf[0] = 65$. Then:

$$\mathcal{N}_{key} = buildScale(k \bmod 12, "major") = \{0, 2, 4, 5, 7, 9, 11, 12, \ldots, 127\}$$
$$\mathcal{N}_{brw} = buildScale(t \bmod 12, "borrowed") = \{2, 4, 5, \mathbf{10}, 14, \ldots, 125\}$$
$$\therefore \mathcal{N}_{all} = \{0, 2, 4, 5, 7, 9, \mathbf{10}, 11, 12, \ldots, 127\}$$

To ensure that borrowed notes are used sparingly, they must be given a cost to use. Let $OffKey$ be the set of notes outside the key and $OffKey_{costs}$ the list of costs associated with each note. The cost for a note will be <*no cost*> or $cost_{OffKey}$ (*DFLT:* <*high cost*>).

$$OffKey = [0, 1, 2, \ldots, 127] \setminus \mathcal{N}_{key}$$
$$\forall \rho \in positions(m)$$
$$OffKey_{costs}[\rho] = \begin{cases} cost_{OffKey} & \text{if } N[\rho] \in OffKey \\ 0 & \text{otherwise} \end{cases} \tag{C.1}$$
$$\text{moreover } \mathcal{C} = \mathcal{C} \cup \sum_{c \in OffKey_{costs}} c$$

G5 *The range of the counterpoint must be consistent with the instrument used.*

This rule is already handled by the creation of the set $\mathcal{N}^{\mathcal{R}} = \mathcal{N} \cap \mathcal{R}$. When N is created its domain is set to $\mathcal{N}^{\mathcal{R}}_{all}$ as seen in the code sample C.2: `*extended-cp-domain` refers to the set $\mathcal{N}^{\mathcal{R}}_{all}$.

G6 *Chromatic melodies are forbidden.*

$$\forall \rho \in positions(m - 2)$$
$$(M_{brut}[\rho] = 1 \wedge M_{brut}[\rho + 1] = 1) \iff \bot \tag{C.2}$$
$$(M_{brut}[\rho] = -1 \wedge M_{brut}[\rho + 1] = -1) \iff \bot$$

G7 *Melodic intervals should be small.*

$$\forall \rho \in positions(m - 1)$$
$$Mdeg_{costs}[\rho] = \begin{cases} cost_{secondMdeg} & \text{if } M[\rho] \in \{0, 1, 2\} \\ cost_{thirdMdeg} & \text{if } M[\rho] \in \{3, 4\} \\ cost_{fourthMdeg} & \text{if } M[\rho] = 5 \\ cost_{tritoneMdeg} & \text{if } M[\rho] = 6 \\ cost_{fifthMdeg} & \text{if } M[\rho] = 7 \\ cost_{sixthMdeg} & \text{if } M[\rho] \in \{8, 9\} \\ cost_{seventhMdeg} & \text{if } M[\rho] \in \{10, 11\} \\ cost_{octaveMdeg} & \text{if } M[\rho] = 12 \end{cases} \tag{C.3}$$
$$\text{moreover } \mathcal{C} = \mathcal{C} \cup \sum_{c \in Mdeg_{costs}} c$$

G8 *3V - The last chord must be composed only of the notes of the harmonic triad.*

$$\forall s \in \{b, c\} : H(s)[0, m-1] \in Cons_{h\_triad} \tag{C.4}$$

G9 *3V - The last chord must have the same fundamental as the one of the scale used throughout the composition.*

$$N(a)[0, m-1] \mod 12 = N(cf)[0, 0] \mod 12 \tag{C.5}$$

Constraints of the First Species

Harmonic Constraints of the First Species

1.H1 *All harmonic intervals must be consonances.*

$$\forall j \in [0, m) \quad H[0, j] \in Cons \tag{C.6}$$

1.H2 *The first harmonic interval must be a perfect consonance.* When dealing with two-part composition:

$$H[0, 0] \in Cons_p \tag{C.7}$$

When dealing with three-part composition: The rule doesn't exist.

1.H3 *The last harmonic intervals must be a perfect consonance.* When dealing with two-part composition:

$$H[0, m-1] \in Cons_p \tag{C.8}$$

When dealing with three-part composition: The rule doesn't exist.

1.H4 *The key tone is tuned according to the first note of the cantus firmus.*

$$\begin{aligned} \neg IsCfB[0, 0] &\implies H[0, 0] = 0 \\ \neg IsCfB[0, m-1] &\implies H[0, m-1] = 0 \end{aligned} \tag{C.9}$$

1.H5 *The voices cannot play the same note at the same time except in the first and last measure.*

$$\forall p_1, p_2 \in \{cf, cp_1, cp_2\}, \text{with} p_1 \neq p_2 \forall j \in \{0, 1, 2, 3\} \forall j \in [1, m-1) \quad N(p_1)[i, j] \neq N(p_2)[i, j] \tag{C.10}$$

1.H6 *Imperfect consonances are preferred to perfect consonances.*

$$\begin{gathered} \forall j \in [0, m) \\ Pcons_{costs}[j] = \begin{cases} cost_{Pcons} & \text{if } H[0, j] \in Cons_p \\ 0 & \text{otherwise} \end{cases} \\ \text{moreover } \mathcal{C} = \mathcal{C} \cup \sum_{c \in Pcons_{costs}} c \end{gathered} \tag{C.11}$$

1.H7 and **1.H8** *The harmonic interval of the penultimate note must be a major sixth or a minor third depending on the cantus firmus pitch. When writing with three voices, the*

harmonic interval must be either a minor third, a perfect fifth, a major sixth or an octave. When dealing with two-part composition:

$$\rho := \max(positions(m)) - 1$$
$$H[\rho] = \begin{cases} 9 & \text{if } IsCfB[\rho] \\ 3 & \text{otherwise} \end{cases} \tag{C.12}$$

where ρ represents the penultimate index of any counterpoint.

When dealing with three-part composition:

$$H[0, m-1] \in \{0, 3, 7, 9\} \tag{C.13}$$

1.H9 *3V - One might use sixths or octaves.* As discussed in **1.H9**, there is no constraint to add for this rule.

1.H10 *3V - Tenths are prohibited in the last chord.*

$$H_{brut}[0, m-1] > 12 \implies H[0, m-1] \notin \{3, 4\} \tag{C.14}$$

1.H11 *3V - Octaves should be preferred over unisons.* As discussed in **1.H11**, there is no constraint to add for this rule.

1.H12 *3V - Last chord cannot include a minor third.*

$$H[0, m-1] \neq 3 \tag{C.15}$$

Melodic Constraints of the First Species

1.M1 *Tritone melodic intervals are forbidden.*

$$\begin{gathered} \forall \rho \in positions(m-1) \\ M[\rho] = 6 \implies Mdeg_{costs}[\rho] = cost_{tritoneMdeg} \end{gathered} \tag{C.16}$$

1.M2 *Melodic intervals cannot exceed a minor sixth interval.*

$$\forall j \in [0, m-1) \quad M[0, j] \leq 8 \tag{C.17}$$

1.M3 *3V - Steps are preferred to skips.* This rule is a duplicate of rule **G7**.

1.M4 *3V - The notes of each part should be as diverse as possible.*

$$\begin{gathered} \forall p \in \{cp_1, cp_2\}, \quad \forall j \in [0, m-1), \quad \forall k \in [j+1, \min(j+3, m-1)] : \\ N(p)[0, j] = N(p)[0, j+k] \iff cost_{variety}[j + m * k] = 1 \end{gathered} \tag{C.18}$$

1.M5 *Each part should stay in its voice range.*

This rule is already covered by the definition of the voice ranges, so no constraint is associated to it.

1.M6 *3V - Melodic intervals cannot be greater or equal to a sixth.* This rule is only a restatement of rule **1.M2**, saying that melodic intervals cannot exceed a minor sixth interval.

Motion Constraints of the First Species

1.P1 *Perfect consonances cannot be reached by direct motion.*

When dealing with two-part composition:

$$\forall j \in [0, m-1) \quad H[0, j+1] \in Cons_p \implies P[0, j] \neq 2 \tag{C.19}$$

When dealing with three-part composition:

$$\begin{aligned} &\forall j \in [0, m-2) : \\ &P[0, j] = 2 \wedge H[0, j+1] \in Cons_p \\ &\iff cost_{\text{direct\_move\_to\_p\_cons}}[j] = 8 \end{aligned} \tag{C.20}$$

1.P2 *Contrary motions are preferred to oblique motions which are preferred to direct motions.*

- $cost_{con}$
  DFLT: <no cost>
- $cost_{obl}$
  DFLT: <low cost>
- $cost_{dir}$
  DFLT: <medium cost>

$$\begin{gathered} \forall j \in [0, m-1) \\ P_{costs}[j] = \begin{cases} cost_{con} & \text{if } P[0, j] = 0 \\ cost_{obl} & \text{if } P[0, j] = 1 \\ cost_{dir} & \text{if } P[0, j] = 2 \end{cases} \\ \text{moreover } \mathcal{C} = \mathcal{C} \cup \sum_{c \in P_{costs}} c \end{gathered} \tag{C.21}$$

1.P3 *At the start of any measure, an octave cannot be reached by the lower voice going up and the upper voice going down more than a third skip.*

$$\begin{gathered} i := \max(\mathcal{B}), \forall j \in [0, m-1) \\ H[0, j+1] = 0 \wedge P[i, j] = 0 \wedge \begin{cases} M_{brut}[i, j] < -4 \wedge IsCfB[i, j] \iff \bot \\ M_{cf}[i, j] < -4 \wedge \neg IsCfB[i, j] \iff \bot \end{cases} \\ \text{where } i \text{ stands for the last beat index in a measure.} \end{gathered} \tag{C.22}$$

1.P4 *3V - Successive perfect consonances should be avoided.*

$$\begin{aligned} &\forall v_1, v_2 \in \{cf, cp_1, cp_2\}, \quad v_1 \neq v_2, \quad \forall j \in [0, m-2) : \\ &(H(v_1, v_2)[0, j] \in Cons_) \wedge (H(v_1, v_2)[0, j+1] \in Cons_p) \\ &\implies Cost_{succ\_p\_cons} = 2 \end{aligned} \tag{C.23}$$

1.P5 *3V - Each part starts distant from the lowest stratum.*

This is not a strict rule but an indication to make easier for the composer to have contrary motions. Since this is neither a requirement nor a preference, it can simply be added as a heuristic for the solver. This is discussed in section 4.1.2, on heuristics.

1.P6 *3V - It is prohibited that all parts move in the same direction.*

To prevent this, we need only look at the motions between the parts and the lowest stratum. If one of their motions is contrary, then it is guaranteed that the three voices will not go in the same direction (because at least one is contrary). The same applies if one of the motions is oblique. The problem arises when all the movements are direct, because this would mean that the three voices are going in the same direction. So it was forbidden to have all motions direct at the same time.

$$\begin{aligned} &\forall j \in [0, m-2): \\ &\bigvee_{p \in \{cf, cp_1, cp_2\}} M(p)[0,j] \neq 2 \end{aligned} \tag{C.24}$$

1.P7 *3V - It is prohibited to use successive ascending sixths on a direct upwards motion.* Either the harmonic interval is not a sixth in any of both positions, or one of them is not moving up.

$$\begin{aligned} &\forall j \in [1, m-1), \quad \forall v_1, v_2 \in \{cf, cp_1, cp_2\} \text{ where } v_1 \neq v_2, \quad \text{sixth} := \{8, 9\}: \\ &(H(v_1, v_2)[0, j-1] \notin \text{sixth}) \vee (H(v_1, v_2)[0, j] \notin \text{sixth}) \\ &\vee M(v_1)[0,j] > 0 \vee M(v_2)[0,j] > 0 \end{aligned} \tag{C.25}$$

Constraints of the Second Species

Harmonic Constraints of the Second Species

2.H1 *Thesis harmonies cannot be dissonant.*

As explained above, there is no constraint to add because it would be a duplicate of rule **1.H1**.

2.H2 *Arsis harmonies cannot be dissonant except if there is a diminution.*

$$\begin{aligned} &\forall j \in [0, m-1) \\ &IsDim[j] = \begin{cases} \top & \text{if } M^2[0,j] \in \{3,4\} \wedge M^1[0,j] \in \{1,2\} \wedge M^1[2,j] \in \{1,2\} \\ \bot & \text{otherwise} \end{cases} \end{aligned} \tag{C.26}$$

$$\forall j \in [0, m-1) \quad \neg IsCons[2,j] \implies IsDim[j] \tag{C.27}$$

2.H3 and **2.H4** *In the penultimate measure the harmonic interval of perfect fifth must be used for the thesis note if possible. Otherwise, a sixth interval should be used instead.*

$$\begin{aligned} &H[0, m-2] \in \{7, 8, 9\} \\ &\therefore penulthesis_{cost} = \begin{cases} cost_{penulthesis} & \text{if } H[0, m-2] \neq 7 \\ 0 & \text{otherwise} \end{cases} \\ &\text{moreover } \mathcal{C} = \mathcal{C} \cup penulthesis_{cost} \end{aligned} \tag{C.28}$$

2.H5 *3V - Major thirds are now allowed in the last chord.*

No need to add a new constraint as this rule is already covered by rules **1.H2 and 1.H3** and **1.H8**.

2.H6 *3V - The half notes must be coherent with respect to the whole notes.*

No need to add a new constraint as this is not an actual rule.

Melodic Constraints of the Second Species

2.M1 *If the two voices are getting so close that there is no contrary motion possible without crossing each other, then the melodic interval of the counterpoint can be an octave leap.*

$$\begin{aligned} &\forall j \in [0, m-1), \forall M_{cf}[j] \neq 0 \\ M[0,j] = 12 \implies &(H_{abs}[0,j] \leq 4) \wedge (IsCfB[j] \iff M_{cf}[j] > 0) \end{aligned} \tag{C.29}$$

2.M2 *Two consecutive notes cannot be the same. When writing a three-part composition, the 4th-to-last, the 3rd-to-last and the 2nd-to-last may be the same.* When dealing with two-part composition:

$$\forall \rho \in positions(m) \quad N[\rho] \neq N[\rho+1] \tag{C.30}$$

When dealing with three-part composition:

$$\begin{aligned} &\forall j \in [1, m-1), \quad j \neq m-2 : \\ &((N[2, j-1] \neq N[0,j]) \wedge (N[0,j] \neq \wedge N[2,j])) \\ &\wedge \\ &((N[2, m-3] \neq N[0, m-2]) \vee (N[0, m-2] \neq N[2, m-2])) \end{aligned} \tag{C.31}$$

Motion Constraints of the Second Species

2.P1 *If the melodic interval of the counterpoint between the thesis and the arsis is larger than a third, then the motion is perceived based on the arsis note.*

$$\forall j \in [0, m-1) \quad P_{real}[j] = \begin{cases} P[2,j] & \text{if } M[0,j] > 4 \\ P[0,j] & \text{otherwise} \end{cases} \tag{C.32}$$

2.P2 *Rule **1.P3** on the battuta octave is adapted such that it focuses on the motion from the note in arsis.* This constraint already had an adapted mathematical notation in the chapter of the first species. Note that this constraint would indeed use P[2] and not P_{real}.

2.P3 *3V - Successive fifths on the downbeat are only allowed when they are separated by a third on the upbeat.*

$$\begin{aligned} &\forall p_1, p_2 \in \{cf, cp_1, cp_2\} \text{ where } p_1 \neq p_2, \quad \forall j \in [0, m-2) : \\ &Cost_{succ\_p\_cons} = \begin{cases} 0 & \text{if } (H(p_1,p_2)[0,j] \notin Cons_p) \vee (H(p_1,p_2)[0,j+1] \notin Cons_p) \\ 0 & \text{if } (H(p_1,p_2)[0,j] = 5) \wedge (H(p_1,p_2)[0,j+1] = 5) \\ & \qquad \wedge (H(p_1,p_2)[2,j] = 3) \vee (H(p_1,p_2)[2,j] = 4) \\ 2 & \text{otherwise} \end{cases} \end{aligned} \tag{C.33}$$

Constraints of the Third Species

Harmonic Constraints of the Third Species

3.H1 *If five notes follow each other by joint degrees in the same direction, then the harmonic interval of the third note must be consonant.*

$$\forall j \in [0, m-1)$$
$$\left(\bigwedge_{i=0}^{3} M[i,j] \leq 2\right) \wedge \left(\bigwedge_{i=0}^{3} M_{brut}[i,j] > 0 \vee \bigwedge_{i=0}^{3} M_{brut}[i,j] < 0\right) \tag{C.34}$$
$$\implies IsCons[2,j]$$

3.H2 *If the third harmonic interval of a measure is dissonant then the second and the fourth interval must be consonant and the third note must be a diminution.*

$$\forall j \in [0, m-1)$$
$$IsCons[2,j] \vee (IsCons[1,j] \wedge IsCons[3,j] \wedge IsDim[j]) \tag{C.35}$$

where $IsDim[j] = \top$ when the 3rd note of the measure j is a diminution.

3.H3 *It is best to avoid the second and third harmonies of a measure to be consonant with a one-degree melodic interval between them.*

$$\forall j \in [0, m-1)$$
$$Cambiata_{costs}[j] = \begin{cases} cost_{Cambiata} & \text{if } IsCons[1,j] \wedge IsCons[2,j] \wedge M[1,j] \leq 2 \\ 0 & \text{otherwise} \end{cases} \tag{C.36}$$

3.H4 *In the penultimate measure, if the cantus firmus is in the upper part, then the harmonic interval of the first note should be a minor third.*

$$\neg IsCfB[m-2] \implies H[0, m-2] = 3 \tag{C.37}$$

3.H5 *3V - The quarter notes must be coherent with respect to the whole notes.* There is no constraint to add for this rule, which isn't really a rule.

3.H6 *3V - If the harmonic triad could not be used on the downbeat, it should be used on the second or third beat.*

$$\begin{aligned} &\forall j \in [0, m-1): \\ &(H[1,j] \notin Cons_{h\_triad}) \wedge (H[2,j] \notin Cons_{h\_triad}) \\ &\iff cost_{harmonic-triad-3rd-species}[j] = 1 \end{aligned} \tag{C.38}$$

Melodic Constraints of the Third Species

3.M1 *Each note and its two beats further peer are preferred to be different.*

$$\forall \rho \in positions(m-2)$$
$$MtwoSame_{costs}[i,j] = \begin{cases} cost_{MtwobSame} & \text{if } M^2[\rho] = 0 \\ 0 & \text{otherwise} \end{cases} \tag{C.39}$$

Motion Constraints of the Third Species

3.P1 *The motion is perceived based on the fourth note.*

This implies that the costs of the motions and the first species constraints on the motions are deducted from $P[3]$.

Constraints of the Fourth Species

Harmonic Constraints of the Fourth Species

4.H1 *Arsis harmonies must be consonant.*

$$\forall j \in [0, m-1) \quad H[2,j] \in Cons \tag{C.40}$$

4.H2 *If the cantus firmus is in the upper part, then no harmonic seventh interval can occur.*

$$\forall j \in [1, m-1) \quad \neg IsCfB[j] \implies H[0,j] \notin \{10, 11\} \tag{C.41}$$

4.H3 and **4.H4** *In the penultimate measure, the harmonic interval of the thesis note must be a major sixth or a minor third depending on the cantus firmus pitch.*

$$H[0, m-2] = \begin{cases} 9 & \text{if } IsCfB[m-2] \\ 3 & \text{otherwise} \end{cases} \tag{C.42}$$

4.H4 *3V - Imperfect consonances are preferred over fifth intervals, which in turn are preferred over octaves.*

This rule is already covered by rule **1.H6** and by the default costs of the search.

Melodic Constraints of the Fourth Species

4.M1 *Arsis half notes should be the same as their next halves in thesis.*

$$\forall j \in [0, m-1) \quad NoSync_{costs} = \begin{cases} cost_{NoSync} & \text{if } M[2,j] \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{C.43}$$

4.M2 *Each arsis note and its two measures further peer are preferred to be different.*

$$\forall j \in [0, m-1)$$
$$MtwomSame_{costs} = \begin{cases} cost_{MtwomSame} & \text{if } N[2,j] = N[2,j+2] \\ 0 & \text{otherwise} \end{cases} \tag{C.44}$$

Motion Constraints of the Fourth Species

4.P1 *Dissonant harmonies must be followed by the next lower consonant harmony.*

$$\forall j \in [1, m-1) \quad \neg IsCons[0,j] \implies M_{brut}[0,j] \in \{-1, -2\} \tag{C.45}$$

4.P2 *If the cantus firmus is in the lower part then no second harmony can be preceded by a unison/octave harmony.*

$$\forall j \in [1, m-1)$$
$$IsCfB[j+1] \implies H[2,j] \neq 0 \wedge H[0,j+1] \notin \{1, 2\} \tag{C.46}$$

4.P3 *3V - Successive fifths are allowed when using ligatures.*

$$\forall v_1, v_2 \in \{cf, cp_1, cp_2\}, \quad \text{with } v_1 \neq v_2, \quad \forall j \in [0, m-2)\text{:}$$
$$Cost_{succ\_p\_cons} = \begin{cases} 0 & \text{if } (H(p_1,p_2)[0,j] \notin Cons_p) \vee (H(p_1,p_2)[0,j+1] \notin Cons_p) \\ 0 & \text{if } (H(p_1,p_2)[0,j] = 5) \wedge (H(p_1,p_2)[0,j+1] = 5) \\ 2 & \text{otherwise} \end{cases} \tag{C.47}$$

4.P4 *3V - Resolving to a fifth is preferred over resolving to an octave.*

This is already covered by the rule **4.H5** (prefer fifths over octaves), since preferring fifths over octaves in *all* cases implies preferring to resolve to a fifth rather than to an octave.

4.P5 *3V - Stationary movement in the bass implies dissonance in the fourth species part.*

$$\begin{aligned} &\forall j \in [0, m-1) : \\ &M(a)[0,j] \neq 0 \iff H[2,j] \in Cons \\ &M(a)[0,j] = 0 \iff H[2,j] \in Dis \end{aligned} \tag{C.48}$$

4.P6 *3V - A note provoking a hidden fifth gets replaced by a rest.*

$$\begin{aligned} &\forall j \in [1, m-1) : \\ &H[0,j] = 7 \wedge P[0,j] = 2 \iff N(0,j-1) = \emptyset \end{aligned} \tag{C.49}$$

Constraints of the Fifth Species

The fifth type has a very specific way of working, which cannot be summarised as easily as the other types, as it requires some additional concepts. Because of this, and because no specific constraints have been added by generalising the fifth species to three voices, we recommend that you read chapter 7 of T. Wafflard's thesis to get the full rules for it.

Appendix D

Code

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**