TECHNISCHE UNIVERSITEIT EINDHOVEN

Department of Mathematics and Computing Science

Master's thesis

An approach to automatic generation and verification of solutions to tonal four-part harmony exercises

by

R.K.P. Pisters

Supervisors: dr. V.A.J. Borghuis
ir. J.L. van Geenen
H.W.M. Maas

Eindhoven, june 2007

Abstract

This master's thesis presents the formal specification of an algorithm capable of checking and completing standard four-part harmony exercises. The intended application of this algorithm is a software tool that allows music students to practice exercises in tonal harmony with the use of a computer. Starting point for developing this approach has been an extensive discussion with the Fontys Conservatory of Tilburg, making the results specifically tailored for this institution's needs. At the same time, the approach is flexible enough to incorporate arbitrary harmony rule sets, and set up in such a way that it contributes to the research activities in automatic music generation employed by Philips Research, Eindhoven.

The thesis describes in detail which steps a computer system needs to follow in order to both check a harmony exercise (and give proper feedback), and generate a solution to an empty exercise. The latter functionality can be used in a didactic application whenever a student making an exercise gets stuck: if the computer system can solve a harmony exercise by itself, it is also able to express a hint to the student, indicating which steps to take in order to come to a solution.

A realization of a software system based on the methods presented in the sequel requires programming, some statistical analysis and a number of evaluation procedures. For such a realization it is recommended to construct the software tool as a plug-in to the music notation software system Finale 2006, which is powerful enough to incorporate all theoretical notions discussed in the thesis, and intuitive enough to make for a user-friendly tool for students in tonal harmony.

Once realized, this software tool would signify an innovative step towards a more computer-aided didactic practice in higher education. Also, the use of didactic software tools can result in a decrease of the number of contact hours while still maintaining a high educational level.

Preface

The work presented in this thesis is the result of a master's project conducted at Philips Research Laboratories Eindhoven, in collaboration with the Fontys Conservatory of Tilburg and the Technische Universiteit Eindhoven. The project concerns the specification of an algorithm capable of both checking completed exercises in tonal four-part harmony, as used for didactic purposes by the Fontys Conservatory, and generating plausible solutions to such exercises.

This thesis is organized as follows. Chapter 1 gives an introduction to the project. It describes the context of the project and explains the didactic process involved in courses in tonal functional harmony, as taught at the Fontys Conservatory, introducing and explaining specific techniques and terminology required to be able to read this thesis. Chapter 2 then gives an elaborate description of the problem. In Chapter 3, we summarize the state of the art in the area of automatic harmonization and discuss how our research has departed from there. The algorithm is then described in Chapters 4 to 6. Chapter 4 introduces the specification primitives that are used in specifying the algorithm, the specification itself is covered by Chapters 5 and 6. Chapter 7 then evaluates the requirements given in Chapter 2. Finally, Chapter 8 comments on the accomplishments achieved and gives some directions for future work.

It is advised to read the chapters and sections in the order in which they appear in the thesis. In order to be able to read Section 1.2 and Chapters 4 up to 6, it is recommended that the reader be able to read notes in treble and bass clef and be familiar with the basics of elementary music theory. The more specialized terms are explained in the glossary in Appendix D. Appendix A contains the details concerning the algorithm specification covered by Chapters 5 en 6, Appendix B contains an overview and categorization of harmony rules, and Appendix C gives a formal grammar which dictates permitted chord progressions.

Acknowledgements

First and foremost, I thank Jurjen van Geenen for creating the wonderful opportunity to do this project, for all the hours invested in supervising my work, and for numerous insights in the area of both computing science and music theory. Working with you has been educational as well as pleasurable. Furthermore, I would like to thank Tijn Borghuis for countless insights, for always keeping a complete overview, and for the meticulousness with which you reviewed my thesis. Thanks also to Henk Maas, for your cooperation in the domain analysis, and your ongoing interest and advice.

Additionally, my gratitude goes to Steffen Pauws and Reinder Haakma at Philips Research, for their continuous support and interest in my project. I also thank Verus Pronk and Jan Korst at Philips Research for helping me with the probability theory of Section 5.2.3.

Finally, I thank my parents Paul and Inge Pisters, and my girlfriend Julia Westendorp, whose never-ending love, support and guidance have been as important as anything else.

Contents

Chapter 1

Introduction

This chapter gives an introduction to the project. Section 1.1 explains the context in which the project was initiated and introduces the institutions involved, Section 1.2 gives an elaborate description of the didactic process in harmony courses.

1.1 Context

All music students at the Fontys Conservatory of Tilburg need to follow a curriculum of several courses in music theory. Due to the rule-based nature of some of these courses, it is a natural idea to try and automate some of the didactic processes involved therein. Moreover, with budgets shrinking and the number of lessons decreasing, some automated didactic support almost becomes a necessity if a high educational level is to be maintained.

Having successfully introduced software tools in courses like ear training, it has been a long-standing wish of the conservatory to employ a similar tool for the classes in tonal harmony. Due to the complexity and the lack of formalization of the rules of harmony, a system fitting the specific needs of the conservatory has as yet not been realized. This project aims for a specification of a system that allows students in tonal harmony to practice exercises with the use of a computer.

Additionally, this project contributes to the research activities of Philips Research Laboratories Eindhoven in the area of automated music composition and analysis, for which a thorough and formal understanding of harmony is indispensable.

1.2 The didactic process

This section describes the didactic process followed in harmony courses at the Fontys Conservatory of Tilburg. In Section 1.2.1, we state what a harmony exercise is, in Section 1.2.2 we explain which steps need to be followed in order to complete a harmony exercise successfully, and Section 1.2.3 we discuss an example exercise. Finally, we make some remarks regarding the nature of the harmony rules. The musical terms that are printed in italics can be found in the glossary in Appendix D.

1.2.1 Description of a harmony exercise

A harmony exercise consists of one *given voice*, either a *bass* or *soprano* line. Some notes in this voice may have been put between brackets; these are *ornamental notes* and need to be treated as such when making the exercise. Some *figuring symbols* may have been filled in already. These must also be respected by the student.

A completed harmony exercise contains the given voice plus three more voices that all sound simultaneously. In case the soprano was given, the three voices to be filled in are all lower: an *alto*, *tenor*, and bass voice. In case of a given bass, these voices are all higher: a tenor, alto and soprano voice. Besides these four voices, a completed exercise contains a sequence of figuring symbols below the staff, indicating the intended chord progression. A correct harmony exercise obeys the rules of harmony given in all sections of the harmony reader [dCM] up to and including the one from which the exercise is taken.

1.2.2 Making a harmony exercise

A student in harmony will be taught to undertake several distinct steps in order to successfully complete a harmony exercise. These steps can be formulated as follows.

Step 1 Analyze the given voice, so as to determine the key characteristics of the exercise.

Step 2 Construct the figuring.

Step 3 Write the *non-given outer voice*.

Step 4 Fill in the *inner voices*.

The next section discusses an example exercise in order to illustrate the terminology and activities given above.

1.2.3 An example exercise

Figure 1.1 gives an empty harmony exercise. One voice has been given, in this case the soprano voice. It is the student's task to fill in the other three voices (the alto, tenor and bass voice) and a corresponding sequence of figuring symbols below the staff. This exercise is taken from the third section of [dCM], which treats the *second inversion* of *main triads*.

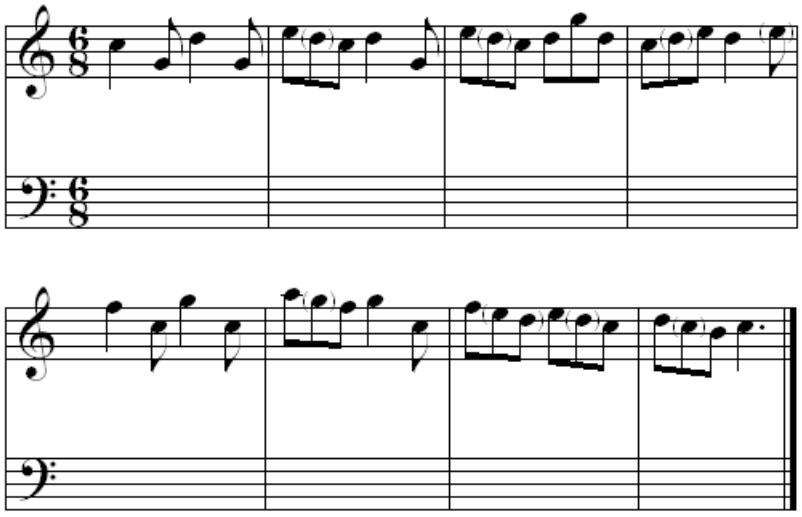

Figure 1.1: An empty exercise

We now fill in this harmony exercise, following the four steps mentioned above.

Step 1 – Analysis of the given voice

The analysis of the given voice needs to identify the key characteristics of this voice, so as to obtain a better understanding of the exercise. The set of characteristics that the analysis needs to cover depends on the section of [dCM] from which the exercise is taken. Every section introduces a new topic, and the exercises typically concern the topic just covered. For instance, the sixth section introduces the chord progression V–VI, and the exercises of this section give considerably more room for applying this chord progression than the exercises in the other sections. A complete overview of the analysis coverage per section is given in Table 1.1. For an explanation of the terms used, see Appendix D.

section	analysis coverage
0	N/A
1	cadences, caesuras, half cadences, harmonic rhythm, hemiolas, ornamental notes
2	cadences, caesuras, half cadences, harmonic rhythm, hemiolas, ornamental notes, sixth chords
3	cadences, caesuras, half cadences, harmonic rhythm, hemiolas, ornamental notes, six-four chords
4	cadences, caesuras, half cadences, harmonic rhythm, hemiolas, ornamental notes, II^6
5	cadences, caesuras, half cadences, harmonic rhythm, hemiolas, ornamental notes, VII^6
6	cadences, caesuras, half cadences, harmonic rhythm, hemiolas, ornamental notes, VI
7	cadences, caesuras, half cadences, harmonic rhythm, hemiolas, ornamental notes, sequences
8	cadences, caesuras, half cadences, harmonic rhythm, hemiolas, ornamental notes

Table 1.1: Analysis coverage overview

Figure 1.2 gives an analysis of the given voice of the example exercise. As we can see, a *half cadence*, a final *cadence*, *ornamental notes*, the *harmonic rhythm*, a *caesura*, and a possibility for a *six-four chord* have been identified. The ornamental notes are those that were already put between brackets in the exercise. In the second half of the fourth measure, we find a half cadence, and the final cadence covers the last two measures. In this final cadence, the *subdominant* comes at the first half of the seventh measure, the *dominant* at the last half of this measure and at the first half of the eighth measure, and the final *tonic* covers the rest of the last measure. The harmonic rhythm has been determined to be such that on average the harmony changes on the first, third, fourth and sixth beat of each measure. Finally, a caesura has been identified in the fourth measure.

Step 2 – Constructing the figuring

Having performed an analysis of the given voice, the figuring can be filled in. The figuring needs to respect the findings of the analysis. For instance, if a half cadence has been identified in the analysis step, it must be figured with V. If at some point a possibility for a six-four chord has been determined, a six-four chord should be preferred to other possible chords at this point.

Figure 1.3 gives a possible figuring of the given voice. The half cadence has been figured with V, and the final cadence with a subdominant, a dominant and a tonic, namely IV_{md}, I^6_4–V, and I, respectively. Also, the ornamental notes have been treated as such and the harmonic rhythm has been followed. Finally, the second half of the seventh measure has indeed been figured with I^6_4.

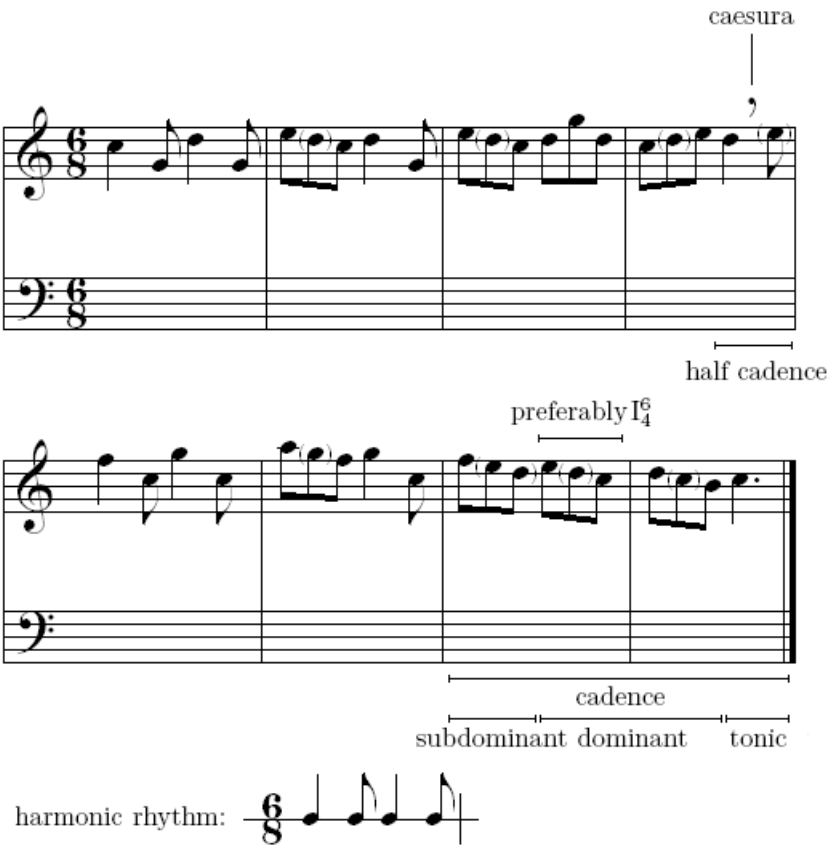

Figure 1.2: An analysis of the given voice

Step 3 – Writing the non-given outer voice

In the third step, the non-given outer voice is filled in. This voice needs to correspond to the figuring symbols determined in the previous step, and it has to obey the *voice leading rules* given in [dCM].

In case of a given soprano, the *pitch classes* of the notes that need to be chosen for the bass voice are already determined by the figuring symbol sequence. For instance, in the example exercise, wherever the figuring is I, a note from pitch class C needs to lie in the bass. In case of IV^6, a note from pitch class A comes in the bass. The only choice is the octave. A possible bass line for the example exercise is given in Figure 1.4.

In case of a given bass, there is a wider choice for the notes in the non-given outer voice (the soprano), as well as an increase need for melodic quality. In this case, the given voice and the figuring need to be analyzed again in order to create a proper soprano voice: if a half cadence has been identified and figured as such at a certain point in the exercise, the second note of the scale needs to lie in the soprano at this point. Also, there are some standard note sequences for the soprano in case some standard pattern is in the bass voice and in the figuring, as exemplified in Figure 1.5. Here, the given bass line is, in case it has been figured with $I–V_4^6–I^6$, usually coupled with the soprano voice given at the right, creating thirds and sixths between the outer voices. This note sequence is to be preferred when writing the soprano voice.

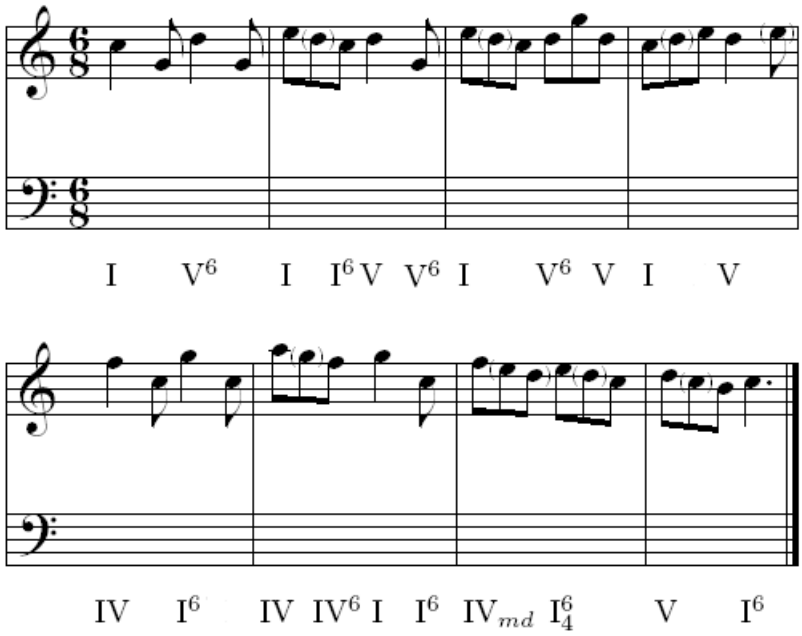

Figure 1.3: A possible figuring

Step 4 – Writing the inner voices

The inner voices must correspond to the figuring symbols chosen in step 2, and they must follow the rules for chord tone duplication and voice leading given in [dCM]. There are certain heuristics to ensure correct inner voices, for instance choosing for either *open* or *close spacing* of chord tones. Concerning chord tone duplications, we can distinguish preferred duplications (e.g. main triads with duplicated root note, one third and one fifth), less preferred duplications (e.g. main triads with one root note, duplicated third and one fifth) and incorrect duplications (e.g. main triads with one root note, three thirds and no fifth). When filling in the inner voices, one should in general always use preferred duplications, and only choose a less preferred duplication to prevent violation of harmony rules that concern voice leading.

Figure 1.6 gives the completely filled in example exercise.

1.2.4 The nature of the rules

In our study of the rules of harmony given in the harmony reader [dCM], we found that there is some variability in the status and strictness of the rules. This has lead to the identification of three rule categories:

- *Strict rules* Strict rules are rules of which a violation constitutes an error and makes the exercise incorrect. These are the most formal rules.
- *Soft rules* Soft rules are rules that make a statement with regard to certain (non-strict) preferences. A violation of a soft rule does not necessarily make a harmony exercise incorrect, but may leave room for improvement.
- *Heuristics* Heuristics are not actual rules, but rather guidelines that can be followed in order to avoid violations of harmony rules.

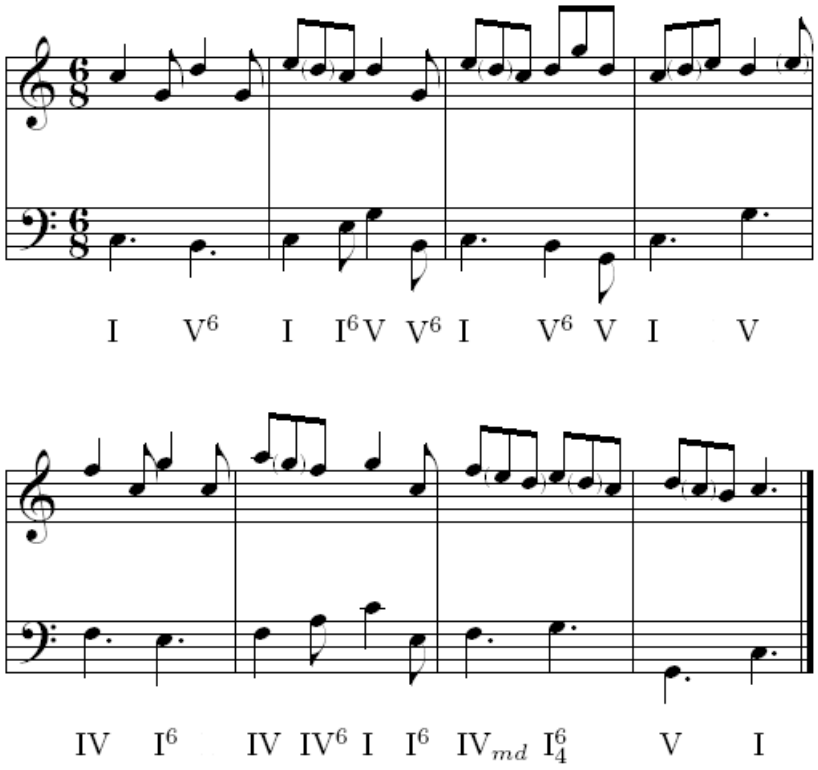

Figure 1.4: A possible bass line

The difference between strict and soft rules is illustrated by Figures 1.7 and 1.8. The strict rule illustrated in Figure 1.7 says that if the notes in two voices are an octave apart, the two following notes in these voices must also not be an octave apart. Here, the bass and and soprano voice both contain an f in the first chord and a g in the second, creating octaves between the outer voices in two consecutive chords. Such parallel octaves are forbidden under all circumstances. The rule illustrated in Figure 1.8 says that the leap in the soprano (g-e) should be continued with a downward movement. In this example, however, the soprano continues its upward movement towards an f, before making a downward leap to a d. The problem that arises as a result of the violation of this rule is dependent on the material that follows, and the downward movement that begins after the f can in this case be seen as a sufficient reason to disregard the rule.

The harmony rules given in the first three sections of the harmony reader [dCM] are given and categorized in Appendix B.

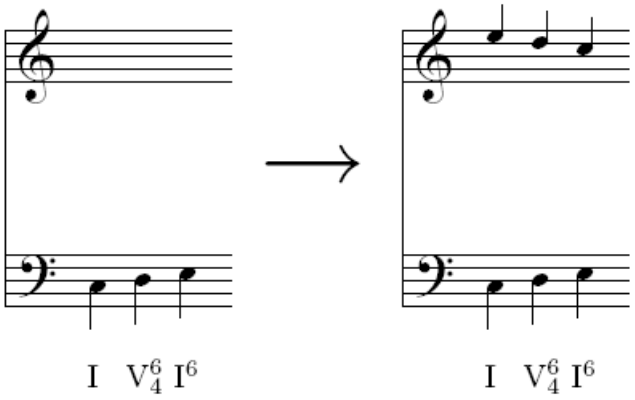

Figure 1.5: A standard way of writing the soprano

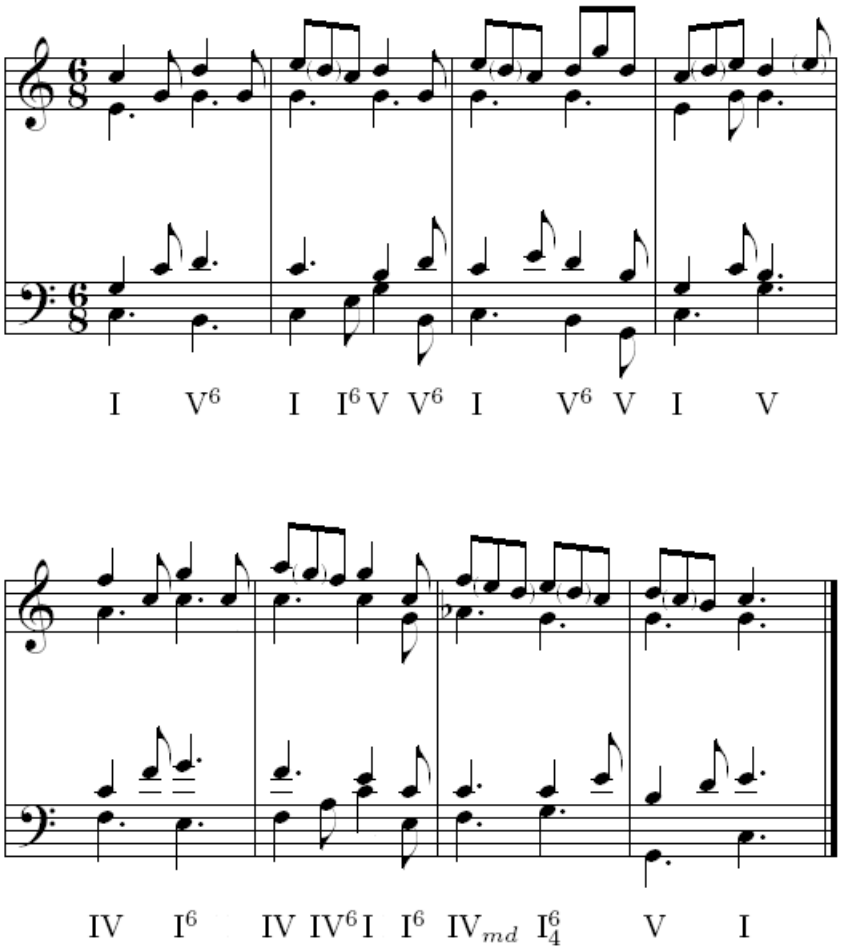

Figure 1.6: The inner voices filled in

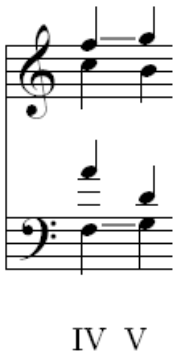

Figure 1.7: Strict rule: Parallel octaves are not allowed.

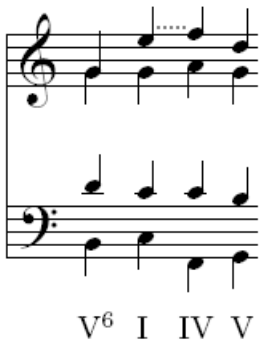

Figure 1.8: Soft rule: After a leap in the soprano, it is preferable to continue in the opposite direction.

Chapter 2

Problem statement

This chapter gives a detailed statement of the problem addressed in this master's project. In short, the project is concerned with the specification of an algorithm to be used in a didactic software tool for harmony courses at the Fontys Conservatory of Tilburg. This tool should be able to check harmony exercises completed by a student, and generate a hint for continuation of a partially completed exercise. Below we give a more detailed problem description. First, Section 2.1 gives an overview of the wishes of the conservatory regarding an eventual software system. Section 2.2 then lists the implications for the underlying algorithm.

2.1 Conservatory wish list

Below we list the requirements on an eventual didactic harmony tool as established in dialogue with the conservatory. First, general requirements are listed; the issues of checking an exercise and generating a hint are dealt with separately in the two sections that follow.

2.1.1 General requirements

In general, we have the following requirements on the system's functionality:

- [G0] The system must be implemented as a plug-in to the music notation software system Finale 2006.
- [G1] The system must be extendable to incorporate different rule sets. These rule sets vary per section of the harmony reader [dCM].

2.1.2 Checking an exercise

In order to check harmony exercises properly, the system is required to have the following functionality:

- [C0] The system must be able to check harmony exercises taken from [dCM] with respect to the rules of the section from which the exercise is taken. More specifically, the system must be able to check (in this order):
  1. the figuring of the given voice,
  2. the outer voices,

3. the inner voices.

- [C1] The system must be able to check, depending on the preferences of the user:
  - only the figuring of the given voice,
  - only the figuring and the outer voices, or
  - figuring, outer voices and inner voices.
- [C2] The system must only check the above aspects if they have been filled in completely.
- [C3] The system must be able to report to the student:
  - if the exercise is correct;
  - if not, what error has been made, and
  - where this error has been made.
- [C4] The exact location of an error must be reported with a textual message and, where possible, graphically in the score.
- [C5] The system must be able to consider the harmony rules as given in [dCM].
- [C6] The system must be able to distinguish strict and soft harmony rules. Corresponding to these types of rules, there are two types of messages that will be shown whenever a rule has been violated. Violation of a strict rule will result in an error message, and violation of a soft rule yields a warning message.
- [C7] The system must be able to generate one error at a time, or all errors at once, depending on the preferences of the user.
- [C8] The system must arrange the error messages in the order in which the errors occur in playback time.

2.1.3 Giving a hint

With regard to giving hints to the student for continuation of a partially completed harmony exercise, the system needs to have the following functionality:

- [H0] In order to give a proper hint, it is required that the student fill in the figuring, the non-given outer voice and the inner voices in this order. If the student does not follow this procedure and asks for a hint, the system will respond indicating that the student is not following the proper approach in order to be allowed to ask for a hint.
- [H1] When giving a hint, the system executes the following steps:
  - first, the exercise filled in so far is checked,
  - if the student has made an error, this will be reported and no hint will be given,
  - if the student has not made an error, the system will try to complete the exercise with the given voice and the material given by the student,
  - if this succeeds, the system gives a hint,
  - if this does not succeed, the system will try to complete the exercise with the given voice and the material given by the student, except for the last position filled in; the last two steps are then reiterated.
- [H2] Depending on the stage the student is currently in, the hint given by the system is a message telling

  - which figuring symbol has to be filled in at the earliest non-filled-in moment,
  - which note in the non-given outer voice has to be filled in at the earliest non-filled-in moment,
  - which notes in the inner voices have to be filled in at the earliest non-filled-in moment, or
  - which figuring symbols or notes filled in so far need to be deleted in order to be able to finish the exercise.

- [H3] The hint must be in accordance with the harmony rules from the chapter of [dCM] selected by the user.

2.2 Algorithm requirements

The system requirements given above say that in case a student asks for a hint, the system will try to complete the exercise with the existing material. This means that the underlying algorithm needs to be capable of generating a solution to an exercise autonomously. We have therefore chosen to distinguish two main functionalities of this algorithm: checking a completed harmony exercise, and generating a correct solution to an empty harmony exercise. A more elaborate statement of the requirements on the algorithm is given by the next two sections.

2.2.1 Checking harmony

In order to be able to properly check a harmony exercise, the underlying algorithm needs to have the following functionality:

- The algorithm's input is a completely filled in four-part harmony exercise.
- The algorithm is able to check the input harmony with respect to an arbitrary rule set. More specifically, it is able to check:
  - only the figuring of the given voice,
  - only the figuring and the outer voices, and
  - figuring, outer voices and inner voices.
- The algorithm must be able to distinguish two rule types, namely strict rules and soft rules. Strict rules are rules that must be obeyed under any circumstances, soft rules are less restrictive and can be violated up to some extent.
- The algorithm's output is a (possibly empty) message log. This log can contain two entry types, namely error and warning messages. For each violation of a rule in the exercise found by the algorithm, the message log contains an entry indicating:
  - the entry type (an error in case the rule is strict, a warning in case the rule is soft)
  - the point in the exercise to which the entry pertains,
  - a message describing the nature of the error or warning.

2.2.2 Generating harmony

In order to generate a correct solution to an empty harmony exercise, the underlying algorithm needs to have the following functionality:

- The algorithm's input is an empty four-part harmony exercise.
- The algorithm's output consists of the given voice of the exercise, supported by four-part harmony and a sequence of figuring symbols that are correct with respect to an arbitrary rule set. More specifically, it is able to:
  - properly analyze the given voice,
  - generate a valid figuring of the given voice, given an analysis,
  - generate a valid non-given outer voice, given a figuring,
  - generate valid inner voices, given two outer voices and a figuring.

Chapter 3

Related work

Although the amount of research conducted in the field of algorithmic music composition is extensive, the number of successful attempts to work towards a solution of the automatic harmonization problem in a satisfactory way with high-quality results is actually quite small. Moreover, we have found no attempts in the literature to develop a theory for the automatic harmonization problem to be used in a didactic setting. Below we discuss some of the most notable attempts to address the problem of automatic music composition in general and automatic harmonization in particular, and discuss these attempts in light of our project. This is done in Section 3.1. Section 3.2 draws some conclusions from this literature study.

3.1 Overview

Since the rules in a harmony textbook such as [dCM] are often stated in a constraint-like fashion, the automatic harmonization problem is often regarded as a constraint satisfaction problem in one way or another. An overview of the attempts to solve the problem as such is given in [PR01]. Pachet and Roy go so far as to say that the problem has been solved, but their results in [PR98] and [PR95] are still far from optimal compared to human harmony solutions. The notion of harmonic rhythm is problematic and there is no distinction between enharmonically equivalent tones. Also, the approach discussed in [PR95] consists of two phases which do not correspond to the four steps in the didactic process given in Chapter 1.2, which we wish to adhere. Although the technique of constraint satisfaction seems to be valuable, the results from the literature are not very encouraging.

A large part of the research done in algorithmic composition focuses on stochastic processes. Particularly, Markov chains have been widely studied for applications in computer music (see e.g. [Xen92], [Cam94]). Although the identification of stochastic elements in music is a quite natural idea, there are several drawbacks to Markov chains which make this technique unsuitable to our project. The most important problem lies in the fact that the use of Markov chains would signify a memoryless advance through a harmony exercise from beginning to end, basing decisions on very little information regarding what came before. As we have explained in Chapter 1.2, our approach to harmonization does not work that way. Instead, we work towards several key moments in the exercise which are identified beforehand. An increase in memory (higher-order Markov chains) rapidly leads to computational problems, and to greater difficulty in the specification of the algorithm's behavior and the validation of the algorithm requirements. Although we can certainly not rule out that the use of Markov chains could be useful to address the automatic harmonization problem, the limited possibilities in this project concerning preliminary research make it doubtful that an approach entirely based on Markov chains would be very productive for this case.

The fact that tonal harmony is up to some extent hierarchically structured has given rise to the idea of using a grammar-based approach to address the automatic harmonization problem. Most notably, Steedman has constructed generative and categorial grammars to make variations on the 12-bar blues (see e.g. [Ste84]). In this technique, there is no given voice for which a suitable accompaniment needs to be constructed; the approach just takes some basic sequence of chords, and uses a grammar to replace some chords by (sequences of) other chords, thus creating a new chord pattern. For our second step of harmonization discussed on page 13, horizontal correctness of figuring symbols (i.e. whether certain chords are allowed to follow certain other chords) can be guaranteed by a grammar that only dictates permitted sequences of chord symbols – we have in fact constructed such a grammar, see Appendix C –, but vertical correctness (i.e. whether certain notes in the given voice can be harmonized by certain chords) cannot. It is conceivable to extend such a grammar to incorporate also the notes in both outer voices, or even all notes in all voices, but this would result in a grammar with so many terminals and production rules that keeping a general overview would become very difficult, and again a validation of the algorithm requirements would be problematic. Again, we don't wish to discard the validity of such approaches altogether, but considering the scope of the current project, a restriction to grammar-based approaches would most likely be inappropriate.

Attempts to generate music using learning systems have yielded results of questionable value, amongst other reasons because of the amount of time needed before these systems start outputting useful results. Although the approaches in this field to harmonization (see e.g. [BT94], [Wid92]) may be interesting from an academic point of view, the absence of encouraging results combined with the fact that such approaches do not work with a clear set of harmony rules, urge us to strive for other techniques for our didactic application. The same argument goes for automatic harmonization based on evolutionary methods (see e.g. [WPPAT99]).

An interesting approach to the automatic harmonization problem is the use of penalty techniques, which make a certain amount of violations of harmony rules possible, as long as this is kept within bounds ([Sch84]). This concept illustrates well the fact that the rules of harmony are often not rigorous, and that sometimes certain rules can be violated in order to prevent violations of other, more strict rules. Using such an approach in our project could be precarious because it would permit that, up to a certain extent, a wrong example is given to a student. In order to prevent this from happening, we need to devise another way of giving shape to the lack of rigorousness of some of the harmony rules.

Of equal interest is the recombination technique developed by D. Cope (see [Cop96]). This method starts from existing pieces the music literature, which are cut into small segments and subsequently recombined to form a new piece of music. Although some are reportedly impressive, the majority of the results is of low quality. It is also not entirely clear how the technique works in detail. Moreover, trying to simulate this approach to address our project would lead to undesired results, since music literature contains many exceptions to textbook harmony rules which ought to be avoided for didactic reasons.

3.2 Concluding remarks

As we have seen in Chapter 1.2, the process of harmonization, as taught at the Fontys Conservatory, is tackled in four distinct steps. We have found no approach in the literature that tries to solve the problem in these four phases. Closest comes [HLZ96], which first lays out a harmonic plan for the accompaniment and only when this has been finished, selects the actual notes. Since our algorithm for automatic checking of a harmony exercise needs to follow the four steps of Chapter 1.2 (see the requirements on the algorithm in Section 2.2), we aim for an algorithm for generating harmony that adheres to these steps also. Furthermore, the intended application of generating hints to students in tonal harmony motivates such an algorithm: most of the time, the algorithm will need to try and complete the exercise, starting with material given by the student. Since the

student is also obliged to follow these four steps, it is convenient if the algorithm follows the same approach so that it will be able to give the best hints possible. For the same reason, it is desirable that the algorithm for generating harmony uses the same rule set as the algorithm that checks harmony.

The algorithm presented in the sequel therefore takes the four steps explained in Chapter 1.2 as a global model. In light of other attempts to solve the automatic harmonization problem, our approach can be seen as a combination of stochastic methods and constraint satisfaction using generate-and-test, but it does constitute a new take on the problem. The specification of the algorithm is done by posing a set of pre- and postconditions in terms of predicate logic. To this end, first some conventions with regard to the specification primitives need to be introduced. This is done in the next chapter.

Chapter 4

Specification primitives

In this chapter, we introduce the formal primitives in order to be able to to make a translation from musical concepts to some mathematical formalism. The formalism used in this chapter and in the next two is based on the guarded command language introduced by E.W. Dijkstra in [Dij75]. The algorithm behavior is specified by stating pre- and postconditions in terms of predicate logic. These chapters take the rules and figuring symbols of the introduction and the first three sections of [dCM] as a starting point.

In the sequel, we define a data type called *Exercise* that contains information of one exercise. In order to be able to define *Exercise*, we first need to introduce some additional data types. This is done in Section 4.1. De definition of *Exercise* is given in Section 4.2. Section 4.3 gives a definition of a data type called *Log*, which is used for constructing the message log when checking a harmony exercise.

4.1 Auxiliary data types

This section defines the data types necessary to define the *Exercise* type, beginning with the data type *Pitch*:

type $Pitch = \textbf{record}\ [\![\ tcd, acc : \mathbb{Z}\ ]\!]$

We represent a note's pitch using the Relative Representation method introduced in [Far90], which requires two values: the Tone Center Displacement (*tcd*) and the Accidental Amount (*acc*). The first value stands for the distance, measured in scale steps, to the root note of the scale in middle c octave. The value of *acc* corresponds to the accidental(s) of the note: 0 meaning natural, 1 means sharp, -2 means double flat, etc. In G major, for instance, $(0, 0)$ represents the first g above middle c, $(1, -1)$ represents the a-flat one second higher, and $(-7, 0)$ represents the g one octave lower.

We use this method rather than the MIDI Note Number Representation (see [Ass01]) because both filling in and checking a harmony exercise require the ability to distinguish between enharmonically equivalent tones. This is needed since the context in which a note appears dictates its spelling. For instance, the leading tone in D major is c-sharp and not d-flat. Using the Relative Representation, these notes are represented by two different notations: $(-1, 0)$ and $(0, -1)$ respectively, whereas with the MIDI Note Number Representation, both notes correspond to the number 61.

type $Meter$ = **record** $[\![\ num, denom : \mathbb{N}\ ]\!]$

Meter is expressed by the data type $Meter$; here *num* (numerator) stands for the number of beats per measure and *denom* (denominator) says which note counts as one beat. In $\frac{3}{8}$ time, $num = 3$ and $denom = 8$.

type Key = **record** $[\![\ whiteKey : \{0..6\}; acc : \mathbb{Z}; mode : \{0..6\}\ ]\!]$

Key is expressed by the data type Key. In this data type, $whiteKey$ equals a value in $\{0..6\}$, denoting the natural note of the root; *acc* denotes the accidental(s) of the root (0 meaning natural, 1 means sharp, 2 means double sharp, -1 means flat, etc.); *mode* is a value in $\{0..6\}$ indicating the mode: 0 corresponds to ionian, 1 to dorian, 2 to phrygian, 3 to lydian, 4 to mixolydian, 5 to aeolian, and 6 to locrian mode. For our current purposes, only 0 and 5 are used (major and minor mode). For instance, the key E-flat minor is represented by $whiteKey = 2$, $acc = -1$, and $mode = 5$.

type $Note$ = **record** $[\![\ ontime : \mathbb{Q}^+ \cup \{0\};\ pitch : Pitch;\ duration : \mathbb{Q}^+\ ]\!]$

Each individual note is represented by three values: *ontime*, *pitch*, and *duration*. These are adapted from the first three entries in a Cope event, introduced by D. Cope in [Cop96]. Regarding the timing convention in the variables *ontime* and *duration*, we define the ontime of a note that comes at beat i of measure number j (both zero-based) to be $j * meter.num + i$. For instance, in $\frac{4}{4}$ time, an eighth note at the second beat of the second measure has $ontime = 5$ and $duration = \frac{1}{2}$.

type $TimeInterval$ = **record** $[\![\ begin : \mathbb{Q}^+ \cup \{0\};\ end : \mathbb{Q}^+\ ]\!]$

We represent time intervals with the data type $TimeInterval$; *begin* and *end* are the two points in time marking the interval.

type $Voice$ = **record** $[\![\ numNotes : \mathbb{N};\ notes : Note[numNotes]\ ]\!]$

A voice is represented by the data type $Voice$, which contains two values: the variable $numNotes$ denotes the number of notes in the certain voice, and the array *notes* consists of all consecutive notes in the voice.

A figuring symbol is a variable of type $FigSym$, which is the set of all possible figuring symbols, plus the undefined symbol $\perp$. This symbol is used for those cases in which it is not possible to choose a valid figuring symbol. For the definition of this data type, see Appendix A.

The approach to generating harmony that we present in the next chapter divides a harmony exercise in several so-called harmony blocks (short periods of time during which the harmony does not change), and determines a suitable harmonization of each harmony block. To this end, we associate with each harmony block a number of possible figuring symbols, using the data type $PosFigSyms$:

type $PosFigSyms = \textbf{record}$
$[\![$
  $numPos : \mathbb{N};$
  $fig : FigSym[numPos];$
  $prob : \mathbb{R}[numPos]$
$]\!]$

In this data type, *numPos* denotes the number of possible figuring symbol for a certain harmony block, fig is the array containing all possible figuring symbols, and *prob* contains the associated probabilities of choice.

We also need to store, for each harmony block, possibilities for notes in the non-given outer voice (ngov). To this end, we use the data type $PosNGOVNotes$:

type $PosNGOVNotes = \textbf{record}$
$[\![$
  $numPos : \mathbb{N};$
  $ngovNotes : Pitch[numPos];$
  $prob : \mathbb{R}[numPos]$
$]\!]$

This data type is similar to $PosFigSyms$: *numPos* again denotes the number of possibilities, *ngovNotes* is the array containing the possible notes for the non-given outer voice, and *prob* contains the associated probabilities.

Possible duplications of chord tones per harmony block are stored in variables of data type $PosDuplications$:

type $PosDuplications = \textbf{record}$
$[\![$
  $numPos : \mathbb{N};$
  $dup : \mathbb{N}[numVoices][numPos]$
$]\!]$

The duplications are stored as arrays in *dup*; *numPos* represents the number of possible duplications.

Ornamental notes are stored in variables of type $OrnamentalNote$:

type $OrnamentalNote = \textbf{record}$ $[\![$ $ornNoteNumber, substitute : \mathbb{N}$ $]\!]$

For each ornamental note, the variable $ornNoteNumber$ contains the note number of the ornamental note of the given voice, the variable *substitute* expresses the number of the note that is to be considered instead of note number $ornNoteNumber$ when figuring the given voice.

4.2 The *Exercise* data type

We can now define the data type *Exercise* as follows:

type $Exercise$ = **record**
|[
$numVoices : \mathbb{N}$;
$voices : Voice[numVoices]$;
$givenVoice : \mathbb{N}$;
$numMeasures : \mathbb{N}$;
$meter : Meter$;
$key : Key$;
$section : \mathbb{N}$;

$cadenceDom$: **set of** $TimeInterval$;
$cadenceSubDom$: **set of** $TimeInterval$;
$cadenceTonic$: **set of** $TimeInterval$;
$caesuras$: **set of** $\mathbb{Q}^+$;
$halfCadences$: **set of** $TimeInterval$;
$harBlock : \mathbb{Q}^+$;
$hemiolas$: **set of set of** $Q^+ \cup \{0\}$;
$ornamentalNotes$: **set of** $OrnamentalNote$;
$sixFourChords$: **set of** $TimeInterval$;

$posFig : PosFigSyms[numMeasures * \frac{meter.num}{harBlock}]$;
$posNGOV : PosNGOVNotes[numMeasures * \frac{meter.num}{harBlock}]$;
$posDup : PosDuplications[numMeasures * \frac{meter.num}{harBlock}]$;
$figSymSeq : FigSym[numMeasures * \frac{meter.num}{harBlock}]$
]|

Here, $numVoices$ stands for the number of voices in the exercise. The voices in *voices* are ordered from low to high: $voices[0]$ denotes the bass and $voices[numVoices-1]$ denotes the soprano voice. A harmony exercise consists typically of four voices. The algorithms presented in the sequel allows an arbitrary number of voices greater than two, except for the generation and checking of notes in the inner voices, which assumes standard four-part harmony. Generalization to an arbitrary number of inner voices requires some extra effort, but is not expected to lead to major difficulties. The number of the given voice, which is either 0 (the bass voice) or $numVoices-1$ (the soprano), is given by the variable *givenVoice*. The length of an exercise is given by $numMeasures$. Meter and key of the exercise are given by the variables *meter* and *key* respectively, and the value of *section* denotes the number of the section of the harmony reader in which this exercise can be found.

The variables *cadenceDom* to *sixFourChords* contain the meta-data of the exercise. *CadenceDom*, *cadenceSubDom*, and *cadenceTonic* contain the numbers of the harmony blocks at which a cadence can be found in the exercise; at harmony blocks *cadenceDom*, a dominant chord has to be used, at *cadenceSubDom*, a subdominant has to be used, and at *cadenceTonic*, a tonic chord must be written, closing the cadence. *Caesuras* contains those points in time right before which a caesura is present in the exercise. The variable $halfCadence$ contains a set of naturals representing the number of the harmony blocks at which a half cadence is present in the exercise. The variable *harBlock*, from harmony block, corresponds to the notion of harmonic rhythm: its value is a rational number representing the minimal chord duration, i.e. the harmony must not change before *harBlock* beats have passed since the previous change of harmony. For instance,

in the example exercise discussed in Section 1.2.3, the value of $harBlock$ must equal 1, since the shortest note duration in the harmonic rhythm (see Figure 1.2) is an eighth note, which has a duration of 1 beat. The variable *hemiolas* contains sets of sets of rationals. These rationals denote the points in time at which the stress amount is affected by a hemiola. For instance, if there are two hemiolas in an exercise, starting at time t and w and affecting the stress levels of time u and v, and x and y respectively, then $hemiolas = \{\{t, u, v\}, \{w, x, y\}\}$. The variable $ornamentalNotes$ contains the array of ornamental notes of the given voice. Finally, $sixFourChords$ contains the time intervals at which preferably a six-four chord is to be written.

The variable $posFig$ is an array of $numMeasures * \frac{meter.num}{harBlock}$ elements. The array contains for each possible harmony block i in exercise e a list of possibilities for figurings ($e.posFig[i].fig$) with associated probabilities of choice ($e.posFig[i].prob$). The array variable $posNGOV$ is similar to $posFig$; it contains for each harmony block i a list of possible notes for the non-given outer voice ($e.posNGOV[i].ngovNotes$), with the associated probability table $e.posNGOV[i].prob$. The variable $posDup$ contains for each harmony block i a list of possible duplications ($e.posDup[i].dup$).

The variable $figSymSeq$ is an array that contains a figuring symbol for each harmony block.

4.3 The *Log* data type

In order to check a harmony exercise, we need one additional data type *Log* to store error and warning messages with proper feedback. A variable of type *Log* represents an error or warning message corresponding to the input harmony that is being checked:

type Log = **record**
⟦
  $type : \{error, warning\}$;
  $harBlock : \mathbb{N}$;
  $message : string$;
⟧

The variable *type* expresses whether the message is an error or a warning message, the value of $harBlock$ denotes the harmony block number to which the message pertains, and the variable *message* contains the actual message. A log for a harmony exercise typically contains more than one message, and therefore has type *Log*[ ].

Chapter 5

Generating harmony

This chapter discusses the approach to the problem of generating harmony, using the types introduced in Chapter 4, taking the rules and figuring symbols of the introduction and the first three sections of [dCM] as a starting point. Section 5.1 gives a breakdown of the problem into three subproblems, to be discussed in Sections 5.2, 5.3, and 5.4.

5.1 Overview

This section gives an overview of the automated process of harmonization, by presenting a breakdown of this process into three subprocesses. Section 5.1.1 explains this breakdown in natural language, Section 5.1.2 gives a formalization with annotated pseudocode.

5.1.1 Description

We divide the problem of generating harmony for a given soprano or bass voice in the following three subproblems:

1. Analyze the given voice and construct a figuring that conforms to the findings of the analysis. The analysis concerns, for the first three sections of the harmony reader, the identification of cadences, caesuras, half cadences, harmonic rhythm, hemiolas, ornamental notes, and notes in the given voice that can be figured with six-four chords. See Table 1.1 for an overview of the elements of this analysis for each section of [dCM].

2. Analyze the given voice and the figuring and write out the non-given outer voice such that it corresponds to the figuring and doesn't violate harmony rules in combination with the given outer voice. Also take the results of the analysis from step 1 into account.

3. Analyze the given voice and the outer voices and fill in the inner voices.

The reason to choose for such a breakdown is twofold. First, we will see that the process of harmonisation involves several subproblems that have such a degree of complexity that some subdivision is needed to tackle them in isolation (e.g. melody writing, analysis of the given voice, determining duplication of chord tones). Dealing with more of these subproblems at the same time would make matters even more complex and less intuitive. We thought it therefore best to subdivide the problem whilst making sure that the resulting subproblems do not overconstrain each other. More importantly, it is a requirement on the system that it be able to generate hints dependent on the activity the student is currently engaged in: if the student is writing the figuring,

the hint should only concern the figuring; if the student is writing the non-given outer voice, the hint will have to be relevant only to the outer voices, and if the student is trying to fill in the inner voices, the hint should only pertain to this activity. This order also corresponds to the activities employed by a human filling in a harmony exercise.

As we can see, these steps correspond closely to the steps taken in the didactic process explained in Chapter 1.2, the only difference being that the first two steps of the didactic process (analysis of the given voice and construction of the figuring) are now combined into one step. The reason for this is that both writing the non-given outer voice and writing the inner voices require a new analysis of the material present in the exercise: when writing the non-given outer voice, the given voice and figuring need to be analyzed, and when writing the inner voices, the figuring and outer voices need to be analyzed. We have already, up to a certain extent, discussed this necessity in Section 1.2.3. Because of this, we chose to design an algorithm that works in three steps of the same shape: first an analysis, followed by the construction of one of the three elements to be written (figuring, non-given outer voice, or inner voices).

5.1.2 Pseudocode

In this section, we give annotated pseudocode for the main approach just presented in terms of the data types presented in Chapter 4.

We use a main function $GenerateHarmony$, which takes an exercise e of type $Exercise$ as argument. This function has as precondition that only the given voice contains notes ($P0$), and as postcondition that either a correct harmonization has been constructed ($P6'$) or that this was unsuccessful and the algorithm failed to construct a suitable harmonization for exercise e, which means that for the first harmony block, no figuring symbol has been chosen ($GenerateHarmony.figSymSeq[0] = \bot$). The assertions $P0..P6$ use the auxiliary predicates $Q0..Q2$. $Q0$ says that for all harmony blocks in exercise e, the figuring is correct, $Q1$ says that the non-given outer voice is correct and $Q2$ says that the inner voices are correct. The function $GenerateHarmony$ calls three functions, to be discussed in the subsequent sections.

$$Q0 : \langle \forall i : 0 \leq i < e.numMeasures * \tfrac{e.meter.num}{e.harBlock} : CorrectFiguring.i \rangle$$

$$Q1 : \langle \forall i : 0 \leq 1 < e.numMeasures * \tfrac{e.meter.num}{e.harBlock} : CorrectNGOV.i \rangle$$

$$Q2 : \langle \forall i : 0 \leq i < e.numMeasures * \tfrac{e.meter.num}{e.harBlock} : CorrectInnerVoices.i \rangle$$

func $GenerateHarmony$(**var** $e : Exercise$) : $Exercise$
|[
  { $P0$ }
  $GenerateHarmony := e$;
  { $P0 \wedge GenerateHarmony = e$ }
  $e := AnalyzeAndMakeFiguring(e)$;
  { $P1$ }
  if $e.figSymSeq[0] \neq \bot \rightarrow$
    { $P2$ }
    $e := AnalyzeAndWriteNGOV(e)$;
    { $P3$ }
    if $e.voices[e.numVoices - 1 - e.givenVoice].numNotes \neq 0 \rightarrow$
      { $P4$ }
      $e := AnalyzeAndWriteInnerVoices(e)$;
      { $P5$ }
      if $e.voices[1].numNotes \neq 0 \rightarrow$

$\{\ P6\ \}$
$GenerateHarmony := e;$
$\{\ P6'\ \}$
$[\!]\ e.voices[1].numNotes = 0 \rightarrow$
skip
fi
$[\!]\ e.voices[e.numVoices - 1 - e.givenVoice].numNotes = 0 \rightarrow$
skip
fi
$[\!]\ e.figSymSeq[0] = \bot \rightarrow$
skip
fi
$\{\ P6' \not\equiv GenerateHarmony.figSymSeq[0] = \bot\ \}$
$]\!]$

$P0 : \langle \forall i : 0 \leq i < e.numVoices : i \neq e.givenVoice \equiv e.voices[i].numNotes = 0 \rangle$

$P1 : P0 \wedge (Q0 \not\equiv e.figSymSeq[0] = \bot)$

$P2 : Q0 \wedge P0$

$P3 : Q0 \wedge \langle \forall i : 1 \leq i < e.numVoices - 1 : e.voices[i].numNotes = 0 \rangle \wedge$
$(Q1 \not\equiv e.voices[e.numVoices - 1 - e.givenVoice].numNotes = 0)$

$P4 : Q0 \wedge Q1 \wedge \langle \forall i : 1 \leq i < e.numVoices - 1 : e.voices[i].numNotes = 0 \rangle$

$P5 : Q0 \wedge Q1 \wedge (Q2 \not\equiv \langle \forall v : 1 \leq v < e.numVoices - 1 : e.voices[v].numNotes = 0 \rangle)$

$P6 : Q0 \wedge Q1 \wedge Q2$

$P6' : P6 \wedge GenerateHarmony = e$

$P0$ says that only the given voice contains a non-empty array of notes. $P1$ says that $P0$ holds, and that either a figuring has been correctly constructed, or that figuring was unsuccessful and the first figuring symbol equals $\bot$. $P3$ says that the figuring is correct, that the inner voices do not contain any notes yet, and that the construction of the non-given voice was either successful ($Q1$) or unsuccessful (the non-given outer voice contains no notes). $P4$ says that the figuring and the non-given outer voice are correct, and that the inner voices have not yet been filled in. $P5$ says that the figuring and the non-given outer voice are correct, and that filling in the inner voices has been either successful ($Q2$) or unsuccessful (the inner voices contain no notes).

Section 5.2 discusses the first step (analysis of given voice and figuring) and thus specifies function *AnalyzeAndMakeFiguring*. Section 5.3 deals with the second step (filling in the non-given outer voice) and specifies function *AnalyzeAndWriteNGOV*. Section 5.4 discusses the third step (filling in the inner voices) and specifies *AnalyzeAndWriteInnerVoices*.

5.2 Analysis and figuring of given voice

This section discusses the process of analyzing and figuring the given voice. Section 5.2.1 explains this process in natural language, Section 5.2.2 gives a formalization with annotated pseudocode.

5.2.1 Description

When figuring a given voice, first this voice needs to be analyzed in order to make some decisions regarding certain key moments in the given voice. Once this analysis has been completed, the figuring can be determined.

Analyzing a given voice means determining the key characteristics of an exercise. Figure 1.1 on page 13 gives an overview of these characteristics per section of [dCM]. For our current purposes, we assume that this information is available in the exercise as meta-data. If, in some future application, this data could be computed, a separate function is available by which this operation can be implemented. The figuring that is to be constructed next must conform to the findings of the analysis. In order to ensure this, the following approach to figuring is taken.

To determine which notes in the given voice correspond to which figuring symbols, we use, per section of the harmony reader, four figuring tables that give, for each note in the given voice, a corresponding set of possible figurings. One of these tables, for the case of soprano notes in the major mode, is given by Table 5.1. The other three tables (the ones for bass notes in the major mode, and soprano and bass notes in the minor mode) are given in Appendix A.2.1. With each entry in the table we associate a probability of choice, such that the probabilities in each row sum up to 1. These probabilities, which are stored in a separate table, are not given here; they will have to be taken from the results of probabilistic analysis of existing, correctly filled in harmony exercises. We choose this approach to guarantee some degree of diversity when generating harmony. For instance, some chords are more likely than others, but always preferring a chord of greater likelihood would result in some chords not being used at all. We have therefore chosen to use a system based on probability values, and thus a statistical analysis becomes inevitable.

Using the specification primitives given in Chapter 4, we define these figuring and probability tables as functions $FigTable$ and $FigProbTable$:

func $FigTable(e : Exercise; n : \mathbb{N}) : FigSym[\,]$

func $FigProbTable(e : Exercise; n : \mathbb{N}) : \mathbb{R}[\,]$

$FigTable$ returns an array of values of type $FigSym$, corresponding to the figuring symbols that are possible with the nth note of the given voice in exercise e. For each value of e and n, the return value of $FigTable(e, n)$ contains the undefined symbol $\perp$. This symbol will be chosen whenever all other figuring symbols violate the harmony rules. $FigProbTable$ returns an array of real numbers, which represent the probability distribution of the figuring symbols in $FigTable$. For each value of e and n, the probability corresponding to the symbol $\perp$ equals 0.

The results of the analysis described above can now be translated to an alteration of the probabilities in the array $e.posFig[i].prob$ (for exercise e and harmony block i), meaning that the choice of certain chords can be made more or less likely. At the point of a half cadence, the probability of choosing V must equal 1, whence the other probabilities (for V^6 and V^6_4) must equal 0. Wherever a six-four chord is preferred, the probability for choosing such a chord must be raised and the probability for choosing other chords must be lowered. In case of a cadence, the progression subdominant-dominant-tonic is secured by setting the probabilities for all other chords to 0. In contrast with a manual analysis of the given voice as given in Figure 1.2 on page 14, caesuras and hemiolas need not be determined yet at this point, since both cannot rule out or change the probability of choice of certain chords. These elements however will play a role when checking figuring rules once the analysis is done and the figuring symbols are being selected.

Constructing a valid figuring for the given voice means selecting a proper figuring symbol for each harmony block i. Starting with the first harmony block ($i = 0$), a figuring symbol is chosen from $e.posFig[i].fig$, obeying the corresponding probability distribution in $e.posFig[i].prob$.

soprano note pitch (tcd mod 7, acc)	possible figurings
(0,0)	I, I^6, I^6_4, IV, IV^6, IV^6_4, $IV^{\emptyset}$, IV_{md}, IV^6_{md}, IV^6_{4md}
(1,0)	V, V^6, V^6_4
(1,1)	V_o, V^6_o
(2,0)	I, I^6_4
(3,0)	IV, IV^6, IV^6_4, IV_{md}, IV^6_{md}, IV^6_{4md}
(3,1)	$IV^{\emptyset}$
(4,0)	V, V^6, V^6_4, V_o, V^6_o, I, I^6, I^6_4
(5,-1)	IV_{md}, IV^6_{4md}
(5,0)	IV, IV^6_4
(6,0)	V, V^6_4

Table 5.1: Figuring table for soprano notes, major mode, major triads in all inversions

Subsequently, the rules concerning the figuring are checked. Apart from the rules mentioned in [dCM], some rules will have to be added to ensure that the figuring generated in this stage can indeed be realised without violation of the rules of harmony. For instance, the following two rules prevent illegal (anti-)parallel motion between the outer voices in the chord progression IV–V:

- [C1P0R9] In case of a given soprano, the first note of the scale followed by the second must not be harmonised with IV–V.
- [C1P0R10] In case of a given soprano, the fourth note of the scale followed by the fifth must not be harmonised with IV–V.

Every section of the harmony reader has its own rule set. When starting to generate a solution to a harmony exercise, the rule set (and also figuring symbol set) is used that corresponds to the section in which the exercise can be found.

If the figuring rules hold, the algorithm proceeds to harmony block $i + 1$. If the algorithm gets stuck at harmony block i (i.e. it has tried all possibilities in $e.posFig[i].fig$), it selects the symbol $\perp$ and backtracks to harmony block $i - 1$, restoring the original probability distribution (the distribution from after the analysis) in $e.posFig[i].prob$.

5.2.2 Pseudocode

We now give pseudocode for the figuring process outlined in Section 5.2.1. The main function, $AnalyzeAndMakeFiguring$, takes one argument e of type $Exercise$ and constructs a valid figuring symbol sequence for e (i.e. it selects a valid figuring symbol for each harmony block). Precondition and postcondition of $e := AnalyzeAndMakeFiguring(e)$ for e of type $Exercise$, are given by $P0$ and $P1'$ respectively. $P0$ and $P1$ were discussed on page 34, $P7$ and $P8$ are discussed in Sections 5.2.3 and 5.2.4 respectively.

func $AnalyzeAndMakeFiguring$(**var** $e : Exercise$) : $Exercise$
|[
  { $P0$ }
  $e := InitFigProbTables(e)$;
  { $P0 \wedge P7$ }
  $e := AnalyzeGivenVoice(e)$;
  { $P0 \wedge P8$ }
  $e := MakeFiguring(e, 0)$;
  { $P1$ }

$$
\begin{array}{l}
\quad AnalyzeAndMakeFiguring := e \\
\quad \{\ P1'\ \} \\
]\!]
\end{array}
$$

$$P1' : P1 \wedge AnalyzeAndMakeFiguring = e$$

The function $InitFigProbTables$ initializes the figuring tables and corresponding probability values for each harmony block. $AnalyzeGivenVoice$ analyzes the given voice and alters the probability values for each harmony block accordingly. $MakeFiguring$ then selects the actual figuring symbols. These functions are specified in the next three sections.

5.2.3 InitFigProbTables

This section specifies function $InitFigProbTables$, introduced in the previous section. The function initializes the arrays $e.posFig[i].fig$ and $e.posFig[i].prob$ (i.e. the possible figuring symbols and the corresponding probability distribution) for each harmony block i in exercise e. It is assumed that the value of $e.harBlock$ is given as meta-data. In case this value can be determined algorithmically, it must be computed before executing $InitFigProbTables$ (we recommend using a function $DetermineHarBlock(e)$ as the first statement in $InitFigProbTables$, which determines the harmonic rhythm of exercise e and fills in the value of $e.harBlock$ such that the harmonic rhythm can be accomodated, as explained on page 31). Precondition and postcondition of $e := InitFigProbTables(e)$ for e of type $Exercise$, are given by $P0$ and $P0 \wedge P7$, respectively. $P7$ is given below; it expresses that for all harmony blocks i in exercise e, the figuring tables $e.posFig[i].fig$ and corresponding probability tables $e.posFig[i].prob$ were initialized properly:

$$P7 : \left\langle \forall i : 0 \leq i < e.numMeasures * \frac{e.meter.num}{e.harBlock} : Q3.i \wedge Q4.i \wedge Q5.i \right\rangle$$

The predicate $Q3$ expresses that the number of possible figuring symbols for harmony block i, $e.posFig[i].numPos$, equals the size of the intersection of the figuring tables $FigTable(e, n)$ of all note numbers n in the given voice that occur in harmony block i. In case n is an ornamental note, the figuring table of the substitute for n needs to be chosen instead of that of n. Consider for example the melody with ornamental notes in Figure 5.1. A harmonisation that treats the ornamental notes as such is given in Figure 5.2. The given voice of this exercise e contains the notes $n_0..n_8$, and we have n_1, n_3, n_4, n_6, n_7 as ornamental notes. The substitute of n_1 is n_0, n_2 substitutes n_3, n_5 substitutes n_4, and n_8 substitutes both n_6 and n_7. The definition of $Q3$ can be found in Appendix A.2.1.

The predicate $Q4$ expresses that the array of possible figuring symbols for harmony block i, $e.posFig[i].fig$, equals the the intersection of the figuring tables $FigTable(e, n)$ of all note numbers n in the given voice that occur in harmony block i. Ornamental notes are dealt with as above. $Q4$ can also be found in Appendix A.2.1.

The predicate $Q5$ reflects the initialization of the probability values in $e.posFig[i].prob$ for all harmony blocks i. Since it is possible that a harmony block contains more than one note in the given voice, a mechanism has to be devised to maintain a sensible probability distribution in those cases. To this end, we make the following remarks.

Suppose we have two different probability distributions X and Y. The distributions have domains D_1 and D_2 respectively. From these probability distributions, we construct a new distribution Z, of which the domain equals all values in both D_1 and D_2, and one additional value ξ that corresponds to those elements in X and Y that are not in $D_1 \cap D_2$. We can now define distribution Z as follows:

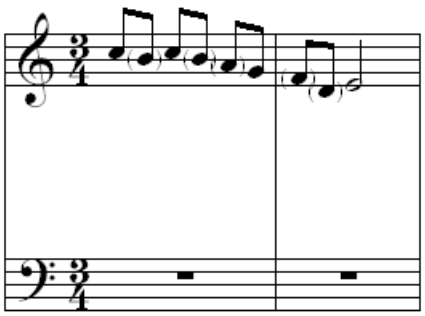

Figure 5.1: A given soprano with ornamental notes

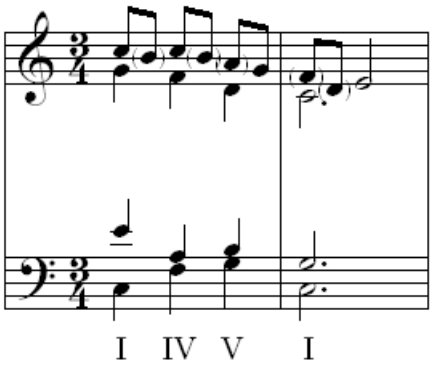

Figure 5.2: A possible harmonization

$$\mathbb{P}(Z=i) = \begin{cases} \mathbb{P}(X=i)\cdot\mathbb{P}(Y=i) & , \quad i \in D_1 \cap D_2 \\ 1 - \sum_{j \in D_1 \cap D_2} \mathbb{P}(X=j)\cdot\mathbb{P}(Y=j) & , \quad i = \xi \end{cases}$$

The intended new probability distribution is however not $\mathbb{P}(Z=i)$, but $\mathbb{P}(Z=i \mid Z \neq \xi)$, i.e. $i \in D_1 \cap D_2$:

$$\mathbb{P}(Z=i \mid Z \neq \xi)$$

$$= \{\text{conditional probability}\}$$

$$\frac{\mathbb{P}(Z=i \wedge Z \neq \xi)}{\mathbb{P}(Z \neq \xi)}$$

$$= \{\text{complement}\}$$

$$\frac{\mathbb{P}(Z=i \wedge Z \neq \xi)}{1-\mathbb{P}(Z=\xi)}$$

$$= \{\text{definition of } \mathbb{P}(Z=i)\}$$

$$\frac{\mathbb{P}(X=i)\cdot\mathbb{P}(Y=i)}{\sum_{j \in D_1 \cap D_2} \mathbb{P}(X=j)\cdot\mathbb{P}(Y=j)}$$

Since this can be generalized to an arbitrary number of initial probability distributions, we conclude that the probability for figuring symbol $e.posFig[i].fig[j]$ in harmony block i, which is stored in $e.posFig[i].prob[j]$, equals the product of the probabilities for this figuring symbol in $FigProbTables(e,n)$, for all note numbers n of a note in the given voice in harmony block i,

$n \notin e.ornamentalNotes$, divided by the sum of all products of probabilities for all figuring symbols in $e.posFig[i].fig$. Below we illustrate this idea with an example. The definitions $Q5$ is given in Appendix A.2.1.

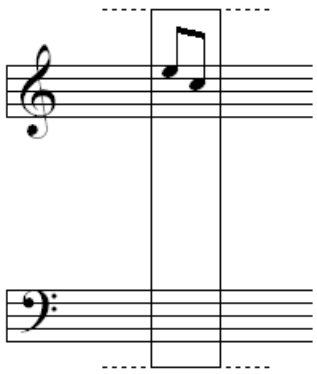

Figure 5.3: Harmony block with two given notes

Example

Consider harmony block i given in Figure 5.3, in the key of C major. This harmony block contains two notes of the given voice of exercise e, $e.voices[e.givenVoice].notes[n_0]$ and $e.voices[e.givenVoice].notes[n_1]$. We assume $n_0, n_1 \notin e.ornamentalNotes$. Function $FigTable$, called with arguments e and n_0, returns [I,I_4^6,$\perp$], and $FigTable(e, n_1)$ returns [I,I^6,I_4^6,IV,IV^6,IV_4^6,$\perp$]. The array of possible figuring symbols for harmony block i, $e.posFig[i].fig$, equals the intersection of $FigTable(e, n_0)$ and $FigTable(e, n_1)$: [I,I_4^6,$\perp$].

The probability distributions for figuring symbols for these two notes are given by $FigProbTable(e, n_0) = [p_0, p_1, p_2]$ and $FigProbTable(e, n_1) = [q_0, q_1, q_2, q_3, q_4, q_5, q_6]$, with $p_2 = q_6 = 0$. The probability for choosing possible figuring symbol $e.posFig[i].fig[j]$ can now be calculated using the probability distribution derived above: the probability for choosing I in harmony block i equals $\frac{p_0 \cdot q_0}{p_0 \cdot q_0 + p_1 \cdot q_2 + p_2 \cdot q_6}$, the probability for I_4^6 equals $\frac{p_1 \cdot q_2}{p_0 \cdot q_0 + p_1 \cdot q_2 + p_2 \cdot q_6}$, and the probability for $\perp$ equals $\frac{p_2 \cdot q_6}{p_0 \cdot q_0 + p_1 \cdot q_2 + p_2 \cdot q_6} = 0$. The sum of these three probabilities equals 1.

5.2.4 AnalyzeGivenVoice

This section specifies function $AnalyzeGivenVoice$, introduced in Section 5.2.2. The function performs an analysis of the given voice and, if necessary, alters the probability values in $e.posFig[i].prob$ for each harmony block i in exercise e. Precondition and postcondition of $e := AnalyzeGivenVoice(e)$ for e of type $Exercise$, are given by $P0 \wedge P7$ and $P0 \wedge P8$, respectively. $P7$ was defined in the previous section, $P8$ is given as follows:

$P8$:
$\big\langle \forall i : 0 \leq i < e.numMeasures * \frac{e.meter.num}{e.harBlock} :$
$\quad \big\langle \sum j : 0 \leq j < e.posFig[i].numPos : e.posFig[i].prob \big\rangle = 1 \wedge$
$\quad i \in HalfCadence(e) \Rightarrow$
$\quad\quad \big\langle \forall j : 0 \leq j < e.posFig[i].numPos : e.posFig[i].prob[j] > 0 \Rightarrow e.posFig[i].fig[j] \in \{\mathrm{I}_4^6, \mathrm{V}\}$
$\quad\quad \big\rangle \wedge$
$\quad i \in SixFour(e) \Rightarrow$

$$
\begin{array}{l}
\quad \langle \exists j : 0 \leq j < e.posFig[i].numPos \wedge e.posFig[i].prob[j] \geq PROBSIXFOUR \wedge \\
\qquad e.posFig[i].fig[j] \in \{\mathrm{I}_4^6, \mathrm{IV}_4^6, \mathrm{V}_4^6\} \\
\quad \rangle \wedge \\
i \in CadenceSubdominant(e) \Rightarrow \\
\quad \langle \forall j : 0 \leq j < e.posFig[i].numPos : e.posFig[i].prob[j] > 0 \Rightarrow \\
\qquad e.posFig[i].fig[j] \in \{\mathrm{IV}, \mathrm{IV}_{md}, \mathrm{IV}_{dm}, \mathrm{IV}^{\emptyset}\} \\
\quad \rangle \wedge \\
i \in CadenceDominant(e) \Rightarrow \\
\quad \langle \forall j : 0 \leq j < e.posFig[i].numPos : e.posFig[i].prob[j] > 0 \Rightarrow e.posFig[i].fig[j] \in \{\mathrm{V}, \mathrm{I}_4^6, \mathrm{V}_o\} \\
\quad \rangle \wedge \\
i \in CadenceTonic(e) \Rightarrow \\
\quad \langle \forall j : 0 \leq j < e.posFig[i].numPos : e.posFig[i].prob[j] > 0 \Rightarrow e.posFig[i].fig[j] \in \{\mathrm{I}, \mathrm{I}_{pic}\} \\
\quad \rangle \\
\rangle
\end{array}
$$

$P8$ uses the functions $HalfCadence$, $SixFour$, $CadenceSubdominant$, $CadenceDominant$, and $CadenceTonic$, all taking an argument of type $Exercise$ and returning a set of naturals. The sets denote the numbers of the harmony blocks at which a half cadence, a possibility for a six-four chord, or a cadence was found in exercise e. $P8$ expresses that if harmony block number i is an element of the return value of $HalfCadence$, only figuring symbols V and I_4^6 must have a probability of choice greater than 0. The cases for six-four chords and cadences are similar. The constant $PROBSIXFOUR$ represents the probability that a six-four chord is actually written when it is preferred. Its value (a rational between 0 and 1) must be determined by statistical analysis of existing, correctly filled in harmony exercises. The functions can be easily specified, using meta-data present in the exercise. For instance, $HalfCadence$ is specified by:

$$
\begin{array}{l}
HalfCadence(e) = \\
\quad \{x \mid 0 \leq x < e.numMeasures * \frac{e.meter.num}{e.harBlock} \wedge \\
\qquad \langle \exists y, z : y \in \mathbb{Q}^+ \cup \{0\} \wedge z \in \mathbb{Q}^+ \wedge [\![\, y, z \,]\!] \in e.halfCadences \wedge y \leq x * e.harBlock < z \rangle \\
\quad \}
\end{array}
$$

$HalfCadence$ returns the number of the harmony blocks that lie in a time interval that is in $e.halfCadences$. The other four functions are similar and specified in Appendix A.2.1. In case this meta-data can be determined computationally, these functions can be used to implement the necessary operations.

5.2.5 MakeFiguring

This section specifies function $MakeFiguring$, introduced in Section 5.2.2. This function selects a figuring symbol for each harmony block i in exercise e from the array $e.posFig[i].fig$, following the probability distribution in $e.posFig[i].prob$. Precondition and postcondition of $e := MakeFiguring(e, 0)$ for e of type $Exercise$, are given by $P0 \wedge P8$ and $P1$ respectively. The postcondition of $e := MakeFiguring(e, i)$ for e of type $Exercise$ and i of type $\mathbb{N}$, is given by:

$$
\begin{array}{l}
P0 \wedge \\
(\\
\quad \langle \forall j : i \leq j < e.numMeasures * \frac{e.meter.num}{e.harBlock} : CorrectFiguring.j \rangle \\
\quad \not\equiv e.figSymSeq[i] = \bot \\
)
\end{array}
$$

This predicate expresses that $P0$ holds and that either for all harmony blocks from i onward, a correct figuring has been constructed, or figuring was unsuccessful and harmony block i has been figured with $\perp$. Filling in $i = 0$ yields $P1$, the postcondition of $MakeFiguring(e, 0)$:

$$
\begin{array}{l}
P0 \wedge \\
(\\
\quad \langle \forall j : 0 \leq j < e.numMeasures * \frac{e.meter.num}{e.harBlock} : CorrectFiguring.j \rangle \\
\quad \not\equiv e.figSymSeq[0] = \perp \\
)
\end{array}
$$

$\equiv$ { definition of $Q0$ }

$P0 \wedge (Q0 \not\equiv e.figSymSeq[0] = \perp)$

$\equiv$ { definition of $P1$ }

$P1$

$Q0$ contains the predicate $CorrectFiguring.i$. This predicate says that some figuring symbol $e.posFig[i].fig[k]$ has been chosen for harmony block i, and that this figuring symbol is correct with respect to the figuring rules. The probability of choosing this figuring symbol thus equals 0 if the figuring rules do not hold, and if these do hold, the probability equals the probability for this figuring symbol given in $e.posFig[i].prob[k]$, divided by the sum S of the probabilities of all correct figuring symbols:

$$
\begin{array}{l}
CorrectFiguring.i : \\
\Big\langle \exists k : 0 \leq k < e.posFig[i].numPos \wedge e.figSymSeq[i] = e.posFig[i].fig[k] \wedge \\
\quad e.figSymSeq[i] \neq \perp \wedge \\
\quad \Big\langle \forall l : 0 \leq l < e.posFig[i].numPos : \\
\qquad \mathbb{P}(k = l) = \begin{cases} 0 & , \ \neg AllRulesFiguring.i \\ \frac{e.posFig[i].prob[l]}{S} & , \ AllRulesFiguring.i \end{cases} \\
\quad \Big\rangle \\
\Big\rangle
\end{array}
$$

$$S : \langle \sum m : e.figSymSeq[i] = e.posFig[i].fig[m] \Rightarrow AllRulesFiguring.i : e.posFig[i].prob[m] \rangle$$

The predicate $AllRulesFiguring.i$ says whether the rules for the figuring hold up till harmony block i:

$$
\begin{array}{l}
AllRulesFiguring.i : \\
RuleC1P0R3.i \wedge RuleC1P0R9.i \wedge RuleC1P0R10.i \wedge \\
(e.section > 0 \Rightarrow \\
\quad RuleC1P1R1.i \wedge RuleC1P1R6.i \wedge RuleC1P1R7.i \wedge RuleC1P1R8.i) \wedge \\
(e.section > 1 \Rightarrow
\end{array}
$$

$RuleC1P2R4.i \wedge RuleC1P2R5.i \wedge RuleC1P2R6.i \wedge RuleC1P2R7.i \wedge$
$RuleC1P2R9.i \wedge RuleC1P2R10.i \wedge RuleC1P2R11.i) \wedge$
$(e.section > 2 \Rightarrow$
$RuleC1P3R1.i \wedge RuleC1P3R2.i \wedge RuleC1P3R5.i \wedge RuleC1P3R7.i \wedge$
$RuleC1P3R8.i \wedge RuleC1P3R9.i)$

Every rule has an associated predicate that expresses whether the rule holds or not for harmony block i. As an example, the predicate for rule C1P1R6, $RuleC1P1R6$, is given in Appendix A.2.1. Although we don't prove this here, we believe the other rules can be written out in a similar way.

5.3 Writing the non-given outer voice

This section addresses the approach to solving the problem of writing the non-given outer voice, given that we constructed a valid figuring. Section 5.3.1 gives a description of this approach in natural language, Section 5.3.2 gives a formalization with annotated pseudocode.

5.3.1 Description

When writing the non-given outer voice, first the given voice and figuring have to be analyzed to find certain key moments, i.e. whether a half cadence can be identified and whether six-four chords were written. Similar to the case of figuring, each harmony block has an associated table of possible notes for the non-given outer voice and a corresponding probability distribution. For exercise e, the possibilities for notes for the non-given outer voice (ngov) in harmony block i are given by the array $e.posNGOV[i].ngovNotes$; the corresponding probability distribution is given by $e.posNGOV[i].prob$.

Selecting notes for the non-given outer voice is similar to selecting figuring symbols in the previous stage of harmonisation: in exercise e, for each harmony block i a possible note for the non-given outer voice is selected from $e.posNGOV[i].ngovNotes$, obeying the probability distribution in $e.posNGOV[i].prob$. This note is added to the non-given outer voice. It is then checked if voice leading rules hold with this new note. If these hold, the algorithm proceeds to the next harmony block, if not, another note is chosen. If all notes have been tried unsuccessfully, the algorithm backtracks to harmony block $i - 1$.

In order to initialize the note and probability tables $e.posNGOV[i].ngovNotes$ and $e.posNGOV[i].prob$, for all harmony blocks i, we need a standard table comparable to $FigTable$ in the previous section. We use the table $NGOVNoteTable$ for the possible notes of the non-given outer voice. Since the probability distribution for possible notes is strongly dependent on the key of the exercise and the average pitch of the given voice, it varies greatly per exercise. Therefore we use a uniform probability distribution for these notes. $NGOVNoteTable$ lists possible notes per figuring symbol. We define this table as a function which takes two arguments: an exercise e and a natural i. The return value is the array of possible pitches for the note in the non-given outer voice for harmony block i, when the figuring symbol is $e.figSymSeq[i]$, the given voice is $e.givenVoice$ and the key is $e.key$. The function $NGOVNoteTable$ is given in Appendix A.2.2.

5.3.2 Pseudocode

Below we give pseudocode for the selection of notes for the non-given outer voice as explained above. The main function, $AnalyzeAndWriteNGOV$, takes one argument e of type $Exercise$ and constructs a valid non-given outer voice for e. Precondition and postcondition of $e := AnalyzeAndWriteNGOV(e)$ for e of type $Exercise$, are given by $P2$ and $P3'$ respectively. $P2$

and $P3$ were discussed on page 34, $P9$ and $P10$ will be discussed in Sections 5.3.3 and 5.3.4 respectively.

func $AnalyzeAndWriteNGOV(e : Exercise) : Exercise$
$[\![$
$\{ P2 \}$
$e := InitNGOVProbTables(e);$
$\{ P2 \wedge P9 \}$
$e := AnalyzeFiguringAndGivenVoice(e);$
$\{ P2 \wedge P10 \}$
$e := WriteNGOV(e, 0);$
$\{ P3 \}$
$AnalyzeAndWriteNGOV := e$
$\{ P3' \}$
$]\!]$

$P3' : P3 \wedge AnalyzeAndWriteNGOV = e$

The function $InitNGOVProbTables$ initializes the note tables for the non-given outer voice and the corresponding probability values for each harmony block. $AnalyzeFiguringAndGivenVoice$ analyzes the given voice and figuring and alters the probability values for each harmony block accordingly. $MakeFiguring$ then selects the actual notes. These functions are specified in the next three sections.

5.3.3 InitNGOVProbTables

This section specifies function $InitNGOVProbTables$, introduced in the previous section. This function initializes the arrays $e.posNGOV[i].ngovNotes$ and $e.posNGOV[i].prob$ (i.e. the possible notes for the non-given outer voice and the corresponding probability distribution) for each harmony block i in exercise e. Precondition and postcondition of $e := InitNGOVProbTables(e)$ for e of type $Exercise$, are given by $P2$ and $P2 \wedge P9$ respectively. $P9$ is given below; it expresses that, for all harmony blocks i,

1. the number of possibilities for notes in the non-given outer voice equals the size of $NGOVNoteTable(e, i)$ (first conjunct),
2. the array that contains these possibilities, $e.posNGOV[i].ngovNotes$, equals $NGOVNoteTable(e, i)$ (second conjunct), and
3. the array that contains the probability distribution of these notes, $e.posNGOV[i].prob$, represent a uniform distribution (third and fourth conjunct).

$P9 :$
$\langle \forall i : 0 \leq i < e.numMeasures * \frac{e.meter.num}{e.harBlock} :$
$e.posNGOV[i].numPos = |NGOVNoteTable[e, i]| \wedge$
$e.posNGOV[i].ngovNotes = NGOVNoteTable[e, i] \wedge$
$\langle \forall j : 1 \leq j < e.posNGOV[i].numPos : e.posNGOV[i].prob[j] = e.posNGOV[i].prob[j-1] \rangle \wedge$
$\langle \sum j : 0 \leq j < e.posNGOV[i].numPos : e.posNGOV[i].prob[j] \rangle = 1$
$\rangle$

The function $NGOVNoteTable$ is given in Appendix A.2.2.

5.3.4 AnalyzeFiguringAndGivenVoice

This section specifies function *AnalyzeFiguringAndGivenVoice*, introduced in Section 5.4.2. This function performs an analysis of the given voice and the figuring and, if necessary, alters the probability values in $e.posNGOV[i].ngovNotes$ and $e.posNGOV[i].prob$ for each harmony block i in exercise e. For the exercises in the third section of the harmony reader [dCM] with a given bass, this analysis needs to identify half cadences (where the second note needs to lie in the soprano) and passages with passing six-four chords (that have a preferred sequence of soprano notes, as illustrated in Figure 1.5 on page 17). In case of a given soprano, no analysis is needed.

Precondition and postcondition of $e := AnalyzeFiguringAndGivenVoice(e)$ for e of type $Exercise$, are given by $P2 \wedge P9$ and $P2 \wedge P10$ respectively. $P9$ was defined in the previous section. Only considering the determination of half cadences, $P10$ is given by:

$$
\begin{array}{l}
P10 : \\
e.givenVoice = 0 \Rightarrow \\
\quad \langle \forall i : i \in HalfCadence(e) \Rightarrow \\
\qquad \langle \forall j : 0 \leq j < e.posNGOV[i].numPos : \\
\qquad\quad \Big(\big(e.posNGOV[i].ngovNotes[j].tcd = 1 \bmod 7 \wedge \\
\qquad\qquad e.posNGOV[i].ngovNotes[j].acc = 0 \big) \\
\qquad\quad \equiv \\
\qquad\quad e.posNGOV[i].prob[j] > 0 \\
\qquad\quad \Big) \\
\qquad \rangle \wedge \\
\qquad \langle \sum j : 0 \leq j < e.posNGOV[i].numPos : e.posNGOV[i].prob[j] \rangle = 1 \\
\quad \rangle
\end{array}
$$

This predicate says that for all harmony blocks at which a half cadence has been identified, the allowed notes for the non-given outer voice need to have a *tcd* of 1 mod 7 and an *acc* of 0, which corresponds to the unaltered second note of the scale. The numbers of these harmony blocks are returned by the function $HalfCadence$ which has been specified on page 41.

The complete definition of of $P10$ (not only for half cadences, but also for passing six-four chords) is given in Appendix A.2.2.

5.3.5 WriteNGOV

This section specifies function $WriteNGOV$, introduced in Section 5.3.2. This function selects a note for each harmony block i in exercise e from the array $e.posNGOV[i].ngovNotes$, following the probability distribution in $e.posNGOV[i].prob$. Precondition and postcondition of $e := WriteNGOV(e, 0)$ for e of type $Exercise$, are given by $P2 \wedge P10$ and $P3$ respectively. The postcondition of $e := WriteNGOV(e, i)$ for e of type $Exercise$ and i of type $\mathbb{N}$ is given by:

$$
\begin{array}{l}
ngov = e.numVoices - 1 - e.givenVoice \wedge \\
Q0 \wedge \langle \forall j : 1 \leq j < e.numVoices - 1 : e.voices[j].numNotes = 0 \rangle \wedge \\
(\langle \forall j : i \leq j < e.numMeasures * \frac{e.meter.num}{e.harBlock} : CorrectNGOV.j \rangle \\
\not\equiv \\
\langle \nexists n : 0 \leq n < e.voices[ngov].numNotes : \\
\quad e.voices[ngov].notes[n].ontime + e.voices[ngov].notes[n].duration \geq i * e.harBlock \rangle \\
)
\end{array}
$$

This predicate says that:

- $Q0$ holds, i.e. a valid figuring has been constructed,
- the inner voices have not yet been filled in, and
- either for all harmony blocks from i onward, a correct non-given outer voice has been constructed, or this was unsuccessful and there is no note sounding in harmony block i.

Filling in $i = 0$ yields $P3$, the postcondition of $AnalyzeAndWriteNGOV(e)$:

$$
\begin{array}{l}
ngov = e.numVoices - 1 - e.givenVoice \wedge \\
Q0 \wedge \langle \forall j : 1 \leq j < e.numVoices - 1 : e.voices[j].numNotes = 0 \rangle \wedge \\
(\langle \forall j : 0 \leq j < e.numMeasures * \frac{e.meter.num}{e.harBlock} : CorrectNGOV.j \rangle \\
\not\equiv \\
\langle \nexists n : 0 \leq n < e.voices[ngov].numNotes : \\
\quad e.voices[ngov].notes[n].ontime + e.voices[ngov].notes[n].duration \geq 0 \rangle \\
)
\end{array}
$$

$\equiv$ { Lemma 1 }

$$
\begin{array}{l}
Q0 \wedge \langle \forall j : 1 \leq j < e.numVoices - 1 : e.voices[j].numNotes = 0 \rangle \wedge \\
(\langle \forall j : 0 \leq j < e.numMeasures * \frac{e.meter.num}{e.harBlock} : CorrectNGOV.j \rangle \\
\not\equiv e.voices[e.numVoices - 1 - e.givenVoice].numNotes = 0 \\
)
\end{array}
$$

$\equiv$ { definition of $Q1$ }

$$
\begin{array}{l}
Q0 \wedge \langle \forall j : 1 \leq j < e.numVoices - 1 : e.voices[j].numNotes = 0 \rangle \wedge \\
(Q1 \not\equiv e.voices[e.numVoices - 1 - e.givenVoice].numNotes = 0)
\end{array}
$$

$\equiv$ { definition of $P3$ }

$P3$

Lemma 1 says that, for any voice v, there is no note in v for which the sum of the ontime and the duration is nonnegative, only if the number of notes in v equals 0. For a proof of Lemma 1, see Appendix A.2.2.

$Q1$ contains the predicate $CorrectNGOV.i$. This predicate says that some note with pitch $e.posNGOV[i].ngovNotes[k]$ has been chosen to sound during harmony block i, and that this note is correct with respect to the rules that concern the outer voices. The probability of choosing this note thus equals 0 if the voice leading rules do not hold, and if these do hold, the probability equals the probability for this note given in $e.posNGOV[i].prob[k]$, divided by the sum S' of the probabilities of all possible correct notes. Predicate $NoteInHarBlock.k.i$ says that a note with pitch $e.posNGOV[i].ngovNotes[k]$ sounds in the non-given outer voice in harmony block i. $CorrectNGOV.i$ is, mutatis mutandis, equal to $CorrectFiguring.i$:

$$
\begin{array}{l}
CorrectNGOV.i : \\
ngov = e.numVoices - 1 - e.givenVoice \wedge \\
\Big\langle \exists k : 0 \leq k < e.posNGOV[i].numPos \wedge NoteInHarBlock.k.i \wedge
\end{array}
$$

$$
\Big\langle \Big\langle \forall l : 0 \leq l < e.posNGOV[i].numPos : \\
\quad \mathbb{P}(k = l) = \begin{cases} 0 & , \ \neg AllRulesNGOV.i \\ \dfrac{e.posNGOV[i].prob[l]}{S'} & , \ AllRulesNGOV.i \end{cases} \\
\Big\rangle \Big\rangle
$$

$$
S' : \Big\langle \sum m : NoteInHarBlock.m.i \Rightarrow AllRulesNGOV.i : e.posNGOV[i].prob[m] \Big\rangle
$$

$$
\begin{array}{l}
NoteInHarBlock.k.i : \\
ngov = e.numVoices - 1 - e.givenVoice \ \wedge \\
\Big\langle \exists n : 0 \leq n < e.voices[ngov].numNotes \ \wedge \\
\quad e.voices[ngov].notes[n].ontime \leq i * e.harBlock \ \wedge \\
\quad e.voices[ngov].notes[n].ontime + \\
\qquad e.voices[ngov].notes[n].duration \geq (i+1) * e.harBlock \ \wedge \\
\quad e.voices[ngov].notes[n].pitch = e.posNGOV[i].ngovNotes[k] \\
\Big\rangle
\end{array}
$$

The predicate *AllRulesNGOV.i* says whether the rules for the non-given outer voice hold up till harmony block i:

$$
\begin{array}{l}
AllRulesNGOV.i : \\
RuleC1P0R2.i \wedge RuleC1P0R4.i \wedge RuleC1P0R5.i \wedge RuleC1P0R8.i \ \wedge \\
(e.section > 0 \Rightarrow \\
\quad RuleC1P1R2SB.i \wedge RuleC1P1R3SB.i \wedge RuleC1P1R4SB.i \ \wedge \\
\quad RuleC1P1R5SB.i \wedge RuleC1P1R9.i \wedge RuleC1P1R10.i \ \wedge \\
\quad RuleC1P1R11.i \wedge RuleC1P1R13.i \wedge RuleC1P1R14 \ \wedge \\
\quad RuleC1P1R15.i \wedge RuleC1P1R16.i \wedge RuleC1P1R17.i) \ \wedge \\
(e.section > 1 \Rightarrow RuleC1P2R2.i \wedge RuleC1P2R12.i) \ \wedge \\
(e.section > 2 \Rightarrow RuleC1P3R3.i \wedge RuleC1P3R4SB.i)
\end{array}
$$

Every rule has an associated predicate that expresses whether the rule holds or not for harmony block i. Rules C1P1R2, C1P1R3, C1P1R4, C1P1R5, and C1P3R4 concern all voices, the suffix "*SB*" in the rule names denotes that only the outer voices are checked. The predicates for rules C1P0R8, C1P1R11, and C1P1R2SB are given in Appendix A.2.2.

5.4 Writing the inner voices

This section addresses the approach to solving the problem of writing the inner voices, given that we have constructed a valid figuring and written a valid non-given outer voice. Section 5.4.1 explains this approach in natural language, Section 5.4.1 gives a formalization with annotated pseudocode.

5.4.1 Description

The biggest problem of filling in the inner voices, is ensuring a correct duplication of chord tones. In case of main triads in root position, each chord preferably consists of two root notes, one third

and one fifth, and if the first two steps have been completed successfully, this usually poses no problems. In some non-standard cases, one can deviate from this rule and write three root notes and one third, or two root notes and two thirds, one root note, one fifth and two thirds, or one root note, one third and two fifths. These are exceptions, however, and they can only occur if no voice leading errors result from them.

Let us now see how this knowledge can be used to fill in the inner voices. We abbreviate the root note, the third and the fifth of a chord by 1, 3, and 5 respectively. Now the standard situation with duplicated root note, one third and one fifth can be denoted by the multiset $[1, 1, 3, 5]$. Having three root notes and one third corresponds to the notation $[1, 1, 1, 3]$.

Determining which figuring symbol in the exercise needs to get which duplication (and thus which multiset) requires an analysis of the figuring. This is again reminiscent of the technique used to construct a figuring: first an initialization part, followed by an analysis and finally the selection of specific notes.

In the initialization part, each harmony block i in the exercise e gets an associated list of duplication multisets, $e.posDup[i].dup$, corresponding to standard duplications only. When considering non-standard duplications, the lists must be extended in the analysis part. The order in which a multiset occurs in a list corresponds to the degree of preference of the duplication represented by this multiset. So in case we are only considering main triads in root position, each list starts with $[1, 1, 3, 5]$, possibly followed by less preferred duplications. In case of the figuring symbol II^6, the corresponding list starts with $[1, 3, 3, 5]$. If we don't want to consider non-standard duplications at all, we can make sure each list only contains one item (meaning $e.posDup[i].numPos$ equals 1, for all harmony blocks i).

When filling in the inner voices, starting from the first harmony block i in the exercise ($i = 0$), first a multiset is chosen from the list (starting with the first element: $e.posDup[i].dup[0]$). Then, notes for the inner voices are chosen, such that the resulting chord corresponds to the multiset. Voice leading rules are checked for these newly selected notes, and if these rules do not hold, other notes are chosen. (With main triads in root position, this approach favors close or open spacing.)

Also here, backtracking is performed in case the algorithm gets stuck, first trying other notes for the inner voices at an earlier point, and if this also fails, trying another multiset from the list of duplications (i.e. double backtracking).

5.4.2 Pseudocode

Below we give pseudocode for the selection of notes for the inner voices as explained above. The main function, $AnalyzeAndWriteInnerVoices$, takes one argument e of type $Exercise$ and constructs valid inner voices for e. Precondition and postcondition of $e := AnalyzeAndWriteInnerVoices(e)$ are given by $P4$ and $P5'$ respectively. $P4$ and $P5$ were discussed on page 34, $P11$ and $P12$ are discussed in Sections 5.4.3 and 5.4.4 respectively.

func $AnalyzeAndWriteInnerVoices(e : Exercise) : Exercise$
|[
  { $P4$ }
  $e := InitMultisets(e)$;
  { $P4 \wedge P11$ }
  $e := AnalyzeFiguringAndOuterVoices(e)$;
  { $P4 \wedge P12$ }
  $e := WriteInnerVoices(e, 0)$;
  { $P5$ }
  $AnalyzeAndWriteInnerVoices := e$

$\{\ P5'\ \}$
$]\!]$

$P5' : P5 \wedge AnalyzeAndWriteInnerVoices = e$

The function $InitMultisets$ initializes the dupication multisets for each harmony block. $AnalyzeFiguringAndOuterVoices$ analyzes the figuring and outer voices and alters the duplication multisets for each harmony block accordingly. $WriteInnerVoices$ selects the actual notes. These functions are specified in the next three sections.

5.4.3 InitMultisets

This section specifies function $InitMultisets$, introduced in the previous section. This function initializes the array $e.posDup[i].dup$ (i.e. the possible chord duplication multisets) for each harmony block i in exercise e. For main triads in all inversions (the third section of [dCM]), the postcondition of $InitMultisets$ is given by:

$P11 : RuleC1P1R0 \wedge RuleC1P2R0 \wedge RuleC1P2R1 \wedge RuleC1P2R8 \wedge RuleC1P3R0$

These rules all concern standard chord note duplications. Rule C1P1R0 says that main triads in root position must have a duplicated root note, one third and one fifth:

$RuleC1P1R0 :$
$\langle \forall i : 0 \leq i < e.numMeasures * \frac{e.meter.num}{e.harBlock} :$
$\quad e.figSymSeq[i] \in \{\mathrm{I},\mathrm{I}_{pic},\mathrm{IV},\mathrm{IV}_{md},\mathrm{IV}_{dm},\mathrm{V},\mathrm{V}_o\}$
$\qquad \Rightarrow (e.posDup[i].numPos = 1 \wedge e.posDup[i].dup[0] = [1,1,3,5])$
$\rangle$

Rule C1P2R0 says that in main triads in first inversion, the melody note must be duplicated. In order to determine the melody note, we use function $ChordNote(tcd, acc : \mathbb{N}; f : FigSym) : \{0,1,3,5,7\}$, of which the return value equals 0 if tcd and acc do not represent a chord note of f, 1 if tcd and acc represent the root note of f, 3 in case of the third, 5 in case of the fifth and 7 in case of the seventh.

$RuleC1P2R0 :$
$\langle \forall i : 0 \leq i < e.numMeasures * \frac{e.meter.num}{e.harBlock} :$
$\quad e.figSymSeq[i] \in \{\mathrm{I}^6,\mathrm{IV}^6,\mathrm{IV}^6_{md},\mathrm{IV}^6_{dm},\mathrm{V}^6,\mathrm{V}^6_o\} \Rightarrow$
$\quad (e.posDup[i].numPos = 1 \wedge e.posDup[i].dup[0] =$
$\qquad [1,3,5,$
$\qquad\quad ChordNote($
$\qquad\qquad GetNoteAtHarBlock(e,i,e.numVoices-1).pitch.tcd,$
$\qquad\qquad GetNoteAtHarBlock(e,i,e.numVoices-1).pitch.acc,$
$\qquad\qquad e.figSymSeq[i]$
$\qquad\quad)$
$\qquad]$
$\quad)$
$\rangle$

The function $GetNoteAtHarBlock(e : Exercise; i, voice : \mathbb{N})$ returns the first note in voice *voice* in harmony block i of exercise e: $GetNoteAtHarBlock = e.voices[voice].notes[n]$, where

$$
\begin{array}{l}
0 \leq n < e.voices[voice].numNotes \wedge \\
e.voices[voice].notes[n].ontime + e.voices[voice].notes[n].duration > i * e.harBlock \wedge \\
\langle \nexists n' : 0 \leq n' < e.voices[voice].numNotes : \\
\quad e.voices[voice].notes[n'].ontime < e.voices[voice].notes[n].ontime \wedge \\
\quad e.voices[voice].notes[n'].ontime + e.voices[voice].notes[n'].duration > i * e.harBlock \\
\rangle
\end{array}
$$

The n is chosen in such a way that the note n in voice *voice* keeps sounding until after the start of harmony block i, and that there is no note n' in voice *voice* coming before note n that also still sounds after the start of harmony block i.

Rule C1P2R1 says that the third in V^6 must not be duplicated:

$$
\begin{array}{l}
RuleC1P2R1 : \\
\langle \forall i : 0 \leq i < e.numMeasures * \frac{e.meter.num}{e.harBlock} : \\
\quad e.figSymSeq[i] = \mathrm{V}^6 \\
\qquad \Rightarrow \langle \sum j : 0 \leq j < e.numVoices \wedge e.posDup[i].dup[0][j] = 3 : 1 \rangle = 1 \\
\rangle
\end{array}
$$

Rule C1P2R8 says that in the Italian sixth chord, the fifth must be duplicated:

$$
\begin{array}{l}
RuleC1P2R8 : \\
\langle \forall i : 0 \leq i < e.numMeasures * \frac{e.meter.num}{e.harBlock} : \\
\quad e.figSymSeq[i] = \mathrm{IV}^{\emptyset} \\
\qquad \Rightarrow (e.posDup[i].numPos = 1 \wedge e.posDup[i].dup[0] = [1, 3, 5, 5]) \\
\rangle
\end{array}
$$

Finally, Rule C1P3R0 says that in six-four chords, the fifth must be duplicated:

$$
\begin{array}{l}
RuleC1P3R0 : \\
\langle \forall i : 0 \leq i < e.numMeasures * \frac{e.meter.num}{e.harBlock} : \\
\quad e.figSymSeq[i] \in \{\mathrm{I}_4^6, \mathrm{IV}_4^6, \mathrm{IV}_{4md}^6, \mathrm{IV}_{4dm}^6, \mathrm{V}_4^6\} \Rightarrow \\
\qquad (e.posDup[i].numPos = 1 \wedge e.posDup[i].dup[0] = [1, 3, 5, 5]) \\
\rangle
\end{array}
$$

5.4.4 AnalyzeFiguringAndOuterVoices

This section specifies function *AnalyzeFiguringAndOuterVoices*, introduced in Section 5.4.2. This function analyzes the figuring and outer voices and determines the situations in which a non-standard duplication of chord tones can be used. This results in a possible update of $e.posDup[i].dup$, for each harmony block i in exercise e. Precondition and postcondition of $e := AnalyzeFiguringAndOuterVoices(e)$ for e of type *Exercise*, are given by $P4 \wedge P11$ and $P4 \wedge P12$ respectively. $P11$ was defined in the previous section. $P12$ is given by:

$P12 : RuleC1P2R3 \wedge RuleC1P3R6$

Both rules concern situations in which standard chord tone duplication can be abandoned. Rule C1P2R3 is expressed by predicate $RuleC1P2R3$ and conerns sixth chords, rule C1P3R6 is expressed by $RuleC1P3R6$ and concerns six-four chords. These predicates can be found in Appendix A.2.3.

5.4.5 WriteInnerVoices

This section specifies function $WriteInnerVoices$, introduced in Section 5.4.2. This function handles the selection of duplications and notes for the inner voices. Precondition and postcondition of $e := WriteInnerVoices(e, 0)$ for e of type $Exercise$ are given by $P4 \wedge P12$ and $P5$ respectively. The postcondition of $e := WriteInnervoices(e, i)$ for e of type $Exercise$ and i of type $\mathbb{N}$, is given by:

$$
\begin{array}{l}
Q0 \wedge Q1 \wedge \\
(\\
\quad \langle \forall j : i \leq j < e.numMeasures * \frac{e.meter.num}{e.harBlock} : CorrectInnerVoices.j \rangle \\
\quad \not\equiv \\
\quad \langle \forall v : 1 \leq v < e.numVoices - 1 : \\
\qquad \langle \nexists n : 0 \leq n < e.voices[v].numNotes : \\
\qquad\quad e.voices[v].notes[n].ontime + e.voices[v].notes[n].duration \geq i * e.harBlock \\
\qquad \rangle \\
\quad \rangle \\
)
\end{array}
$$

This predicate says that:

- $Q0$ holds, i.e. a valid figuring has been constructed,
- $Q1$ holds, i.e. a valid non-given outer voice has been constructed,
- either for all harmony blocks from i onward, correct inner voices were constructed, or this was unsuccessful and there is no note sounding in the inner voices in harmony block i.

Filling in $i = 0$ yields $P5$, the postcondition of $AnalyzeAndWriteInnerVoices(e)$:

$$
\begin{array}{l}
Q0 \wedge Q1 \wedge \\
(\\
\quad \langle \forall j : 0 \leq j < e.numMeasures * \frac{e.meter.num}{e.harBlock} : CorrectInnerVoices.j \rangle \\
\quad \not\equiv \\
\quad \langle \forall v : 1 \leq v < e.numVoices - 1 : \\
\qquad \langle \nexists n : 0 \leq n < e.voices[v].numNotes : \\
\qquad\quad e.voices[v].notes[n].ontime + e.voices[v].notes[n].duration \geq 0 \\
\qquad \rangle \\
\quad \rangle \\
)
\end{array}
$$

$\equiv$ { Lemma 1 }

$$
\begin{array}{l}
Q0 \wedge Q1 \wedge \\
(\\
\quad \big\langle \forall j : 0 \leq j < e.numMeasures * \frac{e.meter.num}{e.harBlock} : CorrectInnerVoices.j \big\rangle \\
\quad \not\equiv \\
\quad \big\langle \forall v : 1 \leq v < e.numVoices - 1 : e.voices[v].numNotes = 0 \big\rangle \\
)
\end{array}
$$

$\equiv$ { definition of $Q2$ }

$$Q0 \wedge Q1 \wedge (Q2 \not\equiv \big\langle \forall v : 1 \leq v < e.numVoices - 1 : e.voices[v].numNotes = 0 \big\rangle)$$

$\equiv$ { definition of $P5$ }

$$P5$$

$Q2$ contains the predicate $CorrectInnerVoices.i$. Since the selection of chord duplications and chord notes is not determined by a probability distribution, $CorrectInnerVoices.i$ differs from $CorrectFiguring.i$ and $CorrectNGOV.i$. We furthermore remark that we have to guarantee a correspondence between the notes in the inner voices in harmony block i and the chord sounding in this harmony block, i.e. the notes have to be chord tones. We use predicate $Q10$ to cover this, predicate $Q11$ guarantees that a permitted chord duplication is chosen. $CorrectInnerVoices.i$ can now be given as follows. It says that notes sounding in the inner voices in harmony i satisfy $\neg Q10$ and $\neg Q11$:

$$
\begin{array}{l}
CorrectInnerVoices.i : \\
\big\langle \forall v : 0 \leq v < e.numVoices : \\
\quad \big\langle \forall n : 0 \leq n < e.voices[v].numNotes \wedge \\
\qquad i * e.harBlock \leq e.voices[v].notes[n].ontime < (i+1) * e.harBlock \vee \\
\qquad (e.voices[v].notes[n].ontime < i * e.harBlock \wedge \\
\qquad\quad e.voices[v].notes[n].ontime + e.voices[v].notes[n].duration > i * e.harBlock) \\
\qquad : \neg Q10.i.v.n \wedge \neg Q11.i.v.n \\
\big\rangle \wedge \\
AllRulesInnerVoices.i
\end{array}
$$

$Q10$ evaluates to *true* if and only if note n in voice v is not a chord tone of the figuring symbol in harmony block i:

$$
\begin{array}{l}
Q10.i.v.n : \\
ChordNote(\\
\quad e.voices[v].notes[n].pitch.tcd, \\
\quad e.voices[v].notes[n].pitch.acc, \\
\quad e.figSymSeq[i] \\
) = 0
\end{array}
$$

Q11 evaluates to *true* if and only if a duplication of chord tones is chosen that does not occur in $e.posDup[i].dup$. To this end, it is checked if the multiset of chord notes of this note, and the

other three notes in the other voices sounding simultaneously, is an element of $e.posDup[i].dup$ (i.e. an allowed duplication) or not. $Q11$ is defined in Appendix A.2.3.

The predicate $AllRulesInnerVoices.i$, used in $CorrectInnerVoices.i$, says whether the rules for the inner voices (except for those covered by $InitMultisets$ and $AnalyzeFiguringAndOuterVoices$) hold up till harmony block i:

$$
\begin{aligned}
&AllRulesInnerVoices.i : \\
&RuleC1P0R0.i \wedge RuleC1P0R6.i \wedge RuleC1P0R1.i \wedge RuleC1P0R2.i \wedge RuleC1P0R7.i \wedge \\
&(e.section > 0 \Rightarrow \\
&\quad RuleC1P1R2.i \wedge RuleC1P1R3.i \wedge RuleC1P1R4.i \wedge RuleC1P1R5.i) \wedge \\
&(e.section > 2 \Rightarrow \\
&\quad RuleC1P3R4.i)
\end{aligned}
$$

Every rule has an associated predicate that expresses whether the rule holds or not for harmony block i. Appendix A.2.3 contains the definition of $RuleC1P3R4$, the predicate that expresses rule C1P3R4

Chapter 6

Checking harmony

In this chapter, we present the approach to checking a harmony exercise. Just like the previous chapter, this chapter takes the rules and figuring symbols of the introduction and the first three sections of [dCM] as a starting point. Section 6.1 gives a breakdown of the problem in three subproblems. These subproblems are then discussed in Sections 6.2, 6.3 and 6.4.

6.1 Overview

This section gives an overview of the process of checking harmony, by presenting a breakdown of the problem into three subproblems. Section 6.1.1 explains this breakdown in natural language, Section 6.1.2 gives a formalization with annotated pseudocode.

6.1.1 Description

Given the algorithm requirements given in Section 2.2, the problem of being able to check a harmony exercise must be stated more precisely by being able to check, given a harmony exercise:

- only the figuring of the given voice,
- only the figuring and the outer voices,
- figuring, outer voices and inner voices.

Therefore, the problem breaks down naturally into the following three subproblems:

1. Checking figuring.
2. Checking outer voices.
3. Checking inner voices.

These subproblems are discussed in Sections 6.2, 6.3 and 6.4, respectively.

6.1.2 Pseudocode

In this section, we give pseudocode for the main approach to checking harmony presented above.

We use a main function $CheckHarmony$, which accepts a (completed) exercise e of type $Exercise$ as an argument. Besides e, it has the boolean arguments $checkOuterVoices$ and $checkInnerVoices$ that indicate which aspects of the exercise are to be checked, and an argument of type $Log[\,]$, that represents the initial log for error and warning messages. The function returns a value of type $Log[\,]$, which is the updated log once checking has been completed. It has precondition $true$ and postcondition $P16'$. The function calls three other functions - these are discussed in subsequent sections.

func $CheckHarmony(e : Exercise;$
$\quad checkOuterVoices, checkInnerVoices : \mathbb{B}; log : Log[\,]) : Log[\,];$
$|[$
$\{\ true\ \}$
$log := AnalyzeAndCheckFiguring(e, log);$
$\{\ P13\ \}$
if $checkOuterVoices \rightarrow$
$\quad log := AnalyzeAndCheckOuterVoices(e, log);$
$\quad \{\ P14\ \}$
$\quad$ **if** $checkInnerVoices \rightarrow$
$\quad\quad log := AnalyzeAndCheckInnerVoices(e, log)$
$\quad\quad \{\ P15\ \}$
$\quad [\!]\ \neg checkInnerVoices \rightarrow$ skip
$\quad$ **fi**
$[\!]\ \neg checkOuterVoices \rightarrow$ skip
fi;
$\{\ P16\ \}$
$CheckHarmony := log$
$\{\ P16'\ \}$
$]|$

$P13 : \langle \forall i : 0 \leq i < e.numMeasures * \frac{e.meter.num}{e.harBlock} : LogFiguring.i.log \rangle$

$P14 : P13 \wedge \langle \forall i : 0 \leq i < e.numMeasures * \frac{e.meter.num}{e.harBlock} : LogOuterVoices.i.log \rangle$

$P15 : P14 \wedge \langle \forall i : 0 \leq i < e.numMeasures * \frac{e.meter.num}{e.harBlock} : LogInnerVoices.i.log \rangle$

$P16 :$
$\neg checkOuterVoices \Rightarrow P13 \wedge$
$(checkOuterVoices \wedge \neg checkInnerVoices) \Rightarrow P14 \wedge$
$(checkOuterVoices \wedge checkInnerVoices) \Rightarrow P15$

$P16' : P16 \wedge CheckHarmony = log$

$P13$ says that for each harmony block, the figuring has been checked an the message log has been updated accordingly. $P14$ and $P15$ state the same for the figuring and the outer voices, and the figuring, outer and inner voices, respectively. The predicates $LogFiguring$, $LogOuterVoices$, and $LogInnerVoices$ are defined in Sections 6.2.3, 6.3.3, and 6.4.3 respectively.

Section 6.2 discusses the first step (checking the figuring), and thus specifies function *AnalyzeAndCheckFiguring*. Section 6.3 deals with the second step (checking the outer voices) and specifies function *AnalyzeAndCheckOuterVoices*. Section 6.4 discusses the third step (checking the inner voices) and specifies *AnalyzeAndCheckInnerVoices*.

6.2 Checking figuring

This section discusses the approach to checking the figuring of the given voice. Section 6.2.1 explains this approach in natural language, Section 6.2.2 gives a formalization with annotated pseudocode.

6.2.1 Description

The process of figuring starts with an analysis of the given voice. After the analysis is done, the figuring is filled in, taking the analysis into account and obeying the rules of chord progression. In order to check if this approach has been followed properly, the process of checking figuring also needs to start with an analysis of the given voice. This is done in the same way as in the case of generating harmony. Again, we use the function *InitFigProbTables* to initialize possible figuring symbols and their probabilities for each harmony block in the exercise. The analysis of the given voice, performed in function *AnalyzeGivenVoice*, then updates these probabilities for certain points in the exercise where, according to the analysis, some figuring symbols have a non-standard degree of preference (e.g. if a half cadence has been found, the probability of V in the table at that point must equal 1, and all other probabilities must equal 0). For more details on this process, see Section 5.2.1.

Checking the figuring is then done in two main steps:

1. Checking correspondence between figuring and given voice.
2. Checking figuring rules.

The first step means checking, for each harmony block i in exercise e, if $e.figSymSeq[i] \in e.posFig[i].fig$. In case a figuring has been chosen that does not occur in $e.posFig[i].fig$, an error message is added to the log, saying that this figuring symbol is incorrect. If a figuring symbol with very low probability has been chosen, a warning message is added. The second step is handled by checking a predicate for each rule and updating the log accordingly.

A variable of type *Log*[ ] is maintained that contains all error and warning messages, and the numbers of the harmony blocks at which they occur.

6.2.2 Pseudocode

In this section we give pseudocode for the approach to checking figuring described above. Analyzing the given voice and checking the figuring is done by function *AnalyzeAndCheckFiguring*, which takes the exercise as argument e, and the message log *log* of type *Log*[ ]. Precondition and postcondition of $log := AnalyzeAndCheckFiguring(e, log)$ for *log* of type *Log*[ ] and e of type *Exercise*, are given by *true* and $P13'$ respectively. $P13$ was defined on page 56. The function calls *InitFigProbTables* and *AnalyzeGivenVoice* (see Sections 5.2.3 and 5.2.4 respectively), and *CheckFiguring*, which performs the actual checking of the figuring symbols.

Pseudocode is given by:

func $AnalyzeAndCheckFiguring(e : Exercise; log : Log[\,]) : Log[\,]$
$[\![$
$\{\ true\ \}$
$e := InitFigProbTables(e);$
$\{\ P7\ \}$
$e := AnalyzeGivenVoice(e);$
$\{\ P8\ \}$
$log := CheckFiguring(e, 0, log);$
$\{\ P13\ \}$
$AnalyzeAndCheckFiguring := log$
$\{\ P13'\ \}$
$]\!]$

$P13' : P13 \wedge AnalyzeAndCheckFiguring = log$

The function $CheckFiguring$ is specified in the next section.

6.2.3 CheckFiguring

This section specifies function $CheckFiguring$, introduced in the previous section. This function checks the figuring and updates the message log accordingly. $CheckFiguring$ accepts an exercise e, a harmony block i and a message log log as arguments, and returns an updated version of log. Precondition and postcondition of $log := CheckFiguring(e, 0, log)$ for e of type $Exercise$ and log of type $Log[\,]$, are given by $P8$ and $P13$ respectively. $P8$ was defined in Section 5.2.4, $P13$ is given on page 56. The postcondition of $log := CheckFiguring(e, i, log)$ for e of type $Exercise$, log of type $Log[\,]$, and i of type $\mathbb{N}$, is given by:

$$P17 : \left\langle \forall j : i \leq j < e.numMeasures * \tfrac{e.meter.num}{e.harBlock} : LogFiguring.j.log \right\rangle$$

This predicate says that for all harmony blocks from i onward, the figuring has been checked and the message log has been updated. Filling in $i = 0$ yields $P13$, the postcondition of $CheckFiguring(e, 0, log)$.

Following the approach explained in Section 6.2.1, pseudocode for $CheckFiguring$ is given by:

func $CheckFiguring(e : Exercise; i : \mathbb{N}; log : Log[\,]) : Log[\,]$
$[\![$
$\{\ P8\ \}$
$log := CheckCorrespondenceFiguring(e, i, log);$
$\{\ P8 \wedge LogCorrespondenceFiguring.i.log\ \}$
$log := CheckFiguringRules(e, i, log);$
$\{\ P8 \wedge LogFiguring.i.log\ \}$
if $(i+1) * \frac{e.harBlock}{e.meter.num} < e.numMeasures \rightarrow$
$\quad log := CheckFiguring(e, i+1, log)$
$[\!]\ (i+1) * \frac{e.harBlock}{e.meter.num} = e.numMeasures \rightarrow$ skip
fi;
$\{\ P17\ \}$
$CheckFiguring := log$

$\{\ P17'\ \}$
$]\!]$

$P17' : P17 \wedge CheckFiguring = log$

The function $CheckCorrespondenceFiguring$ checks whether the notes in the given voice correspond to the figuring symbols written (step 1), and $CheckFiguringRules$ checks the figuring rules (step 2). These functions are specified in the next two sections.

CheckCorrespondenceFiguring

The function $CheckCorrespondenceFiguring(e : Exercise; i : \mathbb{N}; log : Log[\ ])$ checks whether the figuring symbol filled in at harmony block i (stored in $e.figSymSeq[i]$) corresponds to the notes in the given voice. To this end, it is checked if figuring symbol $e.figSymSeq[i]$ occurs in the array of possible figuring symbols for harmony block i, $e.posFig[i].fig$. If it does not, an error message is added to log. If $e.figSymSeq[i]$ is a possible figuring symbol for harmony block i with very low probability (i.e. a probability lower than some bound determined by the function $FigProbBound$), a warning message is added to log.

$CheckCorrespondenceFiguring$ accepts an exercise e, a harmony block i and a message log log as arguments, an returns an updated version of log. Precondition and postcondition of $log := CheckCorrespondenceFiguring(e, i, log)$ for log of type $Log[\ ]$, e of type $Exercise$, and i of type $\mathbb{N}$, are given by $P8$ and $P8 \wedge LogCorrespondenceFiguring.i.log$ respectively. $P8$ was defined in Section 5.2.4, $LogCorrespondenceFiguring$ is given by:

$LogCorrespondenceFiguring.i.log :$
$\langle \forall j : 0 \leq j < e.posFig[i].numPos : e.posFig[i].fig[j] \neq e.figSymSeq[i] \rangle$
$\equiv$
$\langle \exists j : 0 \leq j : log[j] =]\![\ error; i * e.harBlock; \text{“note(s) in given voice cannot be figured with ”} \triangleright e.figSymSeq[i]\ ]\!] \rangle$

$\wedge$

$\langle \exists j : 0 \leq j < e.posFig[i].numPos : e.posFig[i].fig[j] = e.figSymSeq[i] \wedge$
$e.posFig[i].prob[j] < FigProbBound(e, i) \rangle$
$\equiv$
$\langle \exists j : 0 \leq j : log[j] =]\![\ warning; i * e.harBlock; e.figSymSeq[i] \triangleright \text{“ may be an unlikely figuring symbol at this point”}\ ]\!] \rangle$

This predicate says that if the figuring symbol written at harmony block i ($e.figSymSeq[i]$) is not allowed (i.e. it is not in the array of possible figuring symbols for harmony block i, $e.posFig[i].fig$), an error message is added to the log, and that if a figuring symbol has been written with a probability lower than $FigProbBound(e, i)$, a warning message is added.

The function $FigProbBound(e, i) : \mathbb{Q}^+$ returns a rational value between 0 and 1 that denotes the bound that determines whether a figuring symbol is considered unlikely or not. We propose to take as this bound a fraction of the inverse of the number of possible figuring symbols for harmony block i:

func $FigProbBound(e : Exercise; i : \mathbb{N}) : \mathbb{Q}^+$

$$[\![\; FigProbBound := FIGPROBBOUND * \frac{1}{e.posFig[i].numPos} \;]\!]$$

The value of the constant $FIGPROBBOUND$ (a rational between 0 and 1) has to result from statistical analysis and evaluation.

CheckFiguringRules

The function $CheckFiguringRules(e : Exercise; i : \mathbb{N}; log : Log[\,])$ checks whether the sequence of figuring symbols $e.figSymSeq[0]..e.figSymSeq[i]$ is correct with respect to the harmony rules that concern the figuring. Error and warning messages are added to log.

$CheckFiguringRules$ accepts an exercise e, a harmony block i and a message log log as arguments, and returns an updated version of log. Precondition and postcondition of $log := CheckFiguringRules(e, i, log)$ for log of type $Log[\,]$, e of type $Exercise$ and i of type $\mathbb{N}$, are given by $P8 \wedge LogCorrespondenceFiguring.i.log$ and $P8 \wedge LogFiguring.i.log$, respectively. $P8$ was defined in Section 5.2.4, $LogFiguring$ is given by:

$$LogFiguring.i.log : LogCorrespondenceFiguring.i.log \wedge LogFiguringRules.i.log$$

$$\begin{aligned}
&LogFiguringRules.i.log : \\
&LogRuleC1P0R3.i.log \wedge LogRuleC1P0R9.i.log \wedge LogRuleC1P0R10.i.log \wedge \\
&(e.section > 0 \Rightarrow \\
&\quad LogRuleC1P1R1.i.log \wedge LogRuleC1P1R6.i.log \wedge LogRuleC1P1R7.i.log \wedge \\
&\quad LogRuleC1P1R8.i.log) \wedge \\
&(e.section > 1 \Rightarrow \\
&\quad LogRuleC1P2R4.i.log \wedge LogRuleC1P2R5.i.log \wedge LogRuleC1P2R6.i.log \wedge \\
&\quad LogRuleC1P2R7.i.log \wedge LogRuleC1P2R9.i.log \wedge LogRuleC1P2R10.i.log \wedge \\
&\quad LogRuleC1P2R11.i.log) \wedge \\
&(e.section > 2 \Rightarrow \\
&\quad LogRuleC1P3R1.i.log \wedge LogRuleC1P3R2.i.log \wedge LogRuleC1P3R5.i.log \wedge \\
&\quad LogRuleC1P3R7.i.log \wedge LogRuleC1P3R8.i.log \wedge LogRuleC1P3R9.i.log)
\end{aligned}$$

Every rule has an associated predicate that maintains the message log. The predicate for rule C1P1R6, $LogRuleC1P1R6.i.log$, is given in Appendix A.3.1.

6.3 Checking outer voices

This section discusses the approach to checking the outer voices. Section 6.3.1 explains this approach in natural language, Section 6.3.2 gives a formalization with annotated pseudocode.

6.3.1 Description

When generating a non-given outer voice, first the given voice and figuring are analyzed, and the construction of the non-given voice is then based on the findings of this analysis. In order to check if this approach has been followed properly, checking the outer voices also needs to start with an analysis of the given voice and the figuring. This is done in the same way as in the case

of generating harmony: the function $InitNGOVProbTables$ initializes the possible notes for the non-given outer voice and their probability distribution for each harmony block, and the function $AnalyzeFiguringAndGivenVoice$ performs the analysis and updates the probability distributions accordingly. For more details on the workings of these two functions, see Section 5.3.1.

Checking the outer voices is then done in two main steps:

1. Checking correspondence between the non-given outer voice and the figuring,
2. Checking voice leading rules for the outer voices.

In the first step we check, for each harmony block i in exercise e, if all pitches in the non-given voice are in the array of allowed pitches for harmony block i, $e.posNGOV[i].ngovNotes$. In case a pitch has been chosen that does not occur in $e.posNGOV[i].ngovNotes$, an error message is added to the log, saying that this pitch is incorrect. If a pitch with very low probability has been chosen, a warning message is added. The second step checks the harmony rules that concern both outer voices, and the rules that concern voice leading of the non-given outer voice only.

A variable of type $Log[\,]$ is maintained, containing the error and warning messages.

6.3.2 Pseudocode

In this section we give pseudocode for the approach to checking the outer voices described above. Analyzing the given voice and figuring, and checking the non-given outer voice is done by function $AnalyzeAndCheckOuterVoices$, which takes the exercise as argument e, and the message log log of type $Log[\,]$. Precondition and postcondition of $log := AnalyzeAndCheckOuterVoices(e, log)$ for log of type $Log[\,]$ and e of type $Exercise$ are given by $P13$ and $P14'$ respectively. $P13$ and $P14$ were defined on page 56. $AnalyzeAndCheckOuterVoices$ calls $InitNGOVProbTables$ and $AnalyzeFiguringAndGivenVoice$ (see Sections 5.3.3 and 5.3.4 respectively), and $CheckOuterVoices$, which performs the actual checking of the outer voices.

Pseudocode is given by:

func $AnalyzeAndCheckOuterVoices(e : Exercise; log : Log[\,]) : Log[\,]$
|[
  { $P13$ }
  $e := InitNGOVProbTables(e)$;
  { $P13 \wedge P9$ }
  $e := AnalyzeFiguringAndGivenVoice(e)$;
  { $P13 \wedge P10$ }
  $log := CheckOuterVoices(e, 0, log)$;
  { $P14$ }
  $AnalyzeAndCheckOuterVoices := log$
  { $P14'$ }
]|

$P14' : P14 \wedge AnalyzeAndCheckOuterVoices = log$

The function $CheckOuterVoices$ is specified in the next section.

6.3.3 CheckOuterVoices

This section specifies function $CheckOuterVoices$, introduced in the previous section. This function checks the outer voices and updates the message log accordingly. $CheckOuterVoices$ accepts an exercise e, a harmony block i and a message log log as arguments, and returns an updated version of log. Precondition and postcondition of $log := CheckOuterVoices(e, 0, log)$ for e of type $Exercise$ and log of type $Log[\,]$, are given by $P13 \wedge P10$ and $P14$ respectively. $P13$ and $P14$ were defined on page 56. The postcondition of $log := CheckOuterVoices(e, i, log)$ for e of type $Exercise$, log of type $Log[\,]$ and i of type $\mathbb{N}$ is given by:

$$P18 : P13 \wedge \langle \forall j : i \leq j < e.numMeasures * \tfrac{e.meter.num}{e.harBlock} : LogOuterVoices.j.log \rangle$$

This predicate says that $P13$ holds (i.e. the figuring has been checked) and that for all harmony blocks from i onward, the outer voices were checked and the message log has been updated. Filling in $i = 0$ yields $P14$, the postcondition of $CheckOuterVoices(e, 0, log)$.

Following the approach explained in Section 6.3.1, pseudocode for $CheckOuterVoices$ is given by:

func $CheckOuterVoices(e : Exercise; i : \mathbb{N}; log : Log[\,]) : Log[\,]$
[|
  { $P10 \wedge P13$ }
  $log := CheckCorrespondenceNonGivenOuterVoice(e, i, log)$;
  { $P10 \wedge LogCorrespondenceOuterVoices.i.log$ }
  $log := CheckVoiceLeadingRulesOuterVoices(e, i, log)$;
  { $P10 \wedge LogOuterVoices.i.log$ }
  if $(i+1) * \frac{e.harBlock}{e.meter.num} < e.numMeasures \rightarrow$
        $log := CheckOuterVoices(e, i+1, log)$
  [] $(i+1) * \frac{e.harBlock}{e.meter.num} = e.numMeasures \rightarrow$ skip
  fi;
  { $P18$ }
  $CheckOuterVoices := log$
  { $P18'$ }
|]

$P18' : P18 \wedge CheckOuterVoices = log$

The function $CheckCorrespondenceNonGivenOuterVoice$ checks whether the notes in the non-given outer voice correspond to the figuring symbols written (step 1), and $CheckVoiceLeadingRulesOuterVoices$ checks the voice leading rules for the outer voices (step 2). These functions are specified in the next two sections.

CheckCorrespondenceNonGivenOuterVoice

The function $CheckCorrespondenceNonGivenOuterVoice(e : Exercise; i : \mathbb{N}; log : Log[\,])$ checks whether the notes for the non-given outer voice filled in at harmony block i correspond to the notes in the given voice. To this end, it is checked if all notes that sound during harmony block i (which does not necessarily mean their ontimes lie in harmony block i; in case they come earlier, the note should still be sounding in harmony block i) occur in the array of possible notes for harmony block

i, $e.posNGOV[i].ngovNotes$. If a note does not occur in this array, an error message is added to log. If a note occurs that has very low probability in $e.posNGOV[i].prob$ (i.e. a probability lower than some bound determined by the function $NGOVProbBound$), a warning message is added to log.

$CheckCorrespondenceNonGivenOuterVoice$ accepts an exercise e, a harmony block i and a message log log as arguments, an returns an updated version of log. Precondition and postcondition of $log := CheckCorrespondenceNonGivenOuterVoice(e, i, log)$ for log of type $Log[\,]$, e of type $Exercise$, and i of type $\mathbb{N}$, are given by $P10 \wedge P13$ and $P10 \wedge LogCorrespondenceOuterVoices.i.log$ respectively. $P10$ is defined in Section 5.3.4 and $P13$ is given on page 56. $LogCorrespondenceOuterVoices$ is given by:

$$
\begin{array}{l}
LogCorrespondenceOuterVoices.i.log : \\
ngov = e.numVoices - 1 - e.givenVoices \wedge \\
\langle \forall n : 0 \leq n < e.voices[ngov].numNotes \wedge \\
\quad i * e.harBlock \leq e.voices[ngov].notes[n].ontime < (i+1) * e.harBlock \vee \\
\quad (e.voices[v].notes[n].ontime < i * e.harBlock \wedge \\
\qquad e.voices[v].notes[n].ontime + e.voices[v].notes[n].duration > i * e.harBlock) : \\
\qquad \langle \forall j : 0 \leq j < e.posNGOV[i].numPos : \\
\qquad\quad e.posNGOV[i].ngovNotes[j] \neq e.voices[ngov].notes[n].pitch \rangle \\
\qquad \equiv \\
\qquad \langle \exists j : 0 \leq j : log[j] = |[\ error; i * e.harBlock; \text{“note ”} \triangleright e.voices[ngov].notes[n].pitch \triangleright \\
\qquad\qquad\qquad\qquad \text{“ in non-given outer voice is incorrect”}\ ]| \rangle \\
\\
\qquad \wedge \\
\\
\qquad \langle \exists j : 0 \leq j < e.posNGOV[i].numPos : \\
\qquad\quad e.posNGOV[i].ngovNotes[j] = e.voices[ngov].notes[n].pitch \wedge \\
\qquad\quad e.posNGOV[i].prob[j] < NGOVProbBound(e, i) \rangle \\
\qquad \equiv \\
\qquad \langle \exists j : 0 \leq j : log[j] = |[\ warning; i * e.harBlock; \text{“note ”} \triangleright e.voices[ngov].notes[n].pitch \triangleright \\
\qquad\qquad\qquad\qquad \text{“ in non-given outer voice may be unlikely at this point”}\ ]| \rangle \\
\rangle
\end{array}
$$

This predicate says that for all notes of the non-given outer voice that sound in harmony block i (i.e. their ontime is in harmony block i, or their ontime is before harmony block i and they still sound in harmony block i), an error message is added to the message log if this note is not allowed (i.e. it does not occur in the array of possible notes for the non-given outer voice in harmony block i, $e.posNGOV[i].ngovNotes$), and a warning message is added if this note has a probability lower than $NGOVProbBound(e, i)$.

The function $NGOVProbBound(e, i) : \mathbb{Q}^+$ returns a rational value between 0 and 1 that denotes the bound that determines whether a note in the non-given outer voice is considered unlikely or not. We propose to take as this bound a fraction of the inverse of the number of possible notes for harmony block i:

$$
\begin{array}{l}
\textbf{func}\ NGOVProbBound(e : Exercise; i : \mathbb{N}) : \mathbb{Q}^+ \\
|[\\
\quad NGOVProbBound := NGOVPROBBOUND * \frac{1}{e.posNGOV[i].numPos} \\
]|
\end{array}
$$

The value of $NGOVPROBBOUND$ (a rational between 0 and 1) has to result from statistical analysis and evaluation.

CheckVoiceLeadingRulesOuterVoices

The function $CheckVoiceLeadingRulesOuterVoices(e : Exercise; i : \mathbb{N}; log : Log[\ ])$ checks whether the non-given outer voice $e.voices[e.numVoices - 1 - e.givenVoice].notes$ up till harmony block i is correct with respect to the harmony rules that concern the voice leading of the outer voices. Error and warning messages are added to log.

$CheckVoiceLeadingRulesOuterVoices$ accepts an exercise e, a harmony block i and a message log log as arguments, and returns an updated version of log. Precondition and postcondition of $log := CheckVoiceLeadingRulesOuterVoices(e, i, log)$ for log of type $Log[\ ]$, e of type $Exercise$ and i of type $\mathbb{N}$, are given by $P10 \wedge LogCorrespondenceOuterVoices.i.log$ and $P10 \wedge LogOuterVoices.i.log$ respectively. $LogCorrespondenceOuterVoices$ is given in the previous section, $P10$ is defined in Section 5.3.4. $LogOuterVoices$ is given by:

$$
\begin{aligned}
&LogOuterVoices.i.log : LogCorrespondenceOuterVoices.i.log \wedge \\
&\quad LogVoiceLeadingRulesOuterVoices.i.log
\end{aligned}
$$

$$
\begin{aligned}
&LogVoiceLeadingRulesOuterVoices.i.log : \\
&LogRuleC1P0R2SB.i.log \wedge LogRuleC1P0R4.i.log \wedge LogRuleC1P0R5.i.log \wedge \\
&LogRuleC1P0R8.i.log \wedge \\
&(e.section > 0 \Rightarrow \\
&\quad LogRuleC1P1R2SB.i.log \wedge LogRuleC1P1R3.i.log \wedge LogRuleC1P1R4.i.log \wedge \\
&\quad LogRuleC1P1R5.i.log \wedge LogRuleC1P1R9.i.log \wedge LogRuleC1P1R10.i.log \wedge \\
&\quad LogRuleC1P1R11.i.log \wedge LogRuleC1P1R13.i.log \wedge LogRuleC1P1R14.i.log \wedge \\
&\quad LogRuleC1P1R15.i.log \wedge LogRuleC1P1R12.i.log \wedge LogRuleC1P1R16.i.log \wedge \\
&\quad LogRuleC1P1R17.i.log) \wedge \\
&(e.section > 1 \Rightarrow LogRuleC1P2R2.i.log \wedge LogRuleC1P2R12.i.log) \wedge \\
&(e.section > 2 \Rightarrow LogRuleC1P3R3.i.log \wedge LogRuleC1P3R4.i.log)
\end{aligned}
$$

Every rule has an associated predicate that maintains the message log. Rules C1P0R2 and C1P1R2 concern all voices, the suffix "SB" in the rule names indicates that here only the outer voices are checked. The predicates for rules C1P0R8 and C1P1R2SB are given in Appendix A.3.2.

6.4 Checking inner voices

This section discusses the approach to checking the inner voices. Section 6.4.1 explains this approach in natural language, Section 6.4.2 gives a formalization with annotated pseudocode.

6.4.1 Description

When generating notes for the inner voices, first the outer voices and figuring are analyzed, and the selection of notes for the inner voices is then based on the findings of this analysis. In order to check if this approach has been followed properly, checking the inner voices also needs to start with an analysis of the outer voices and the figuring. This is done in the same way as in the case of generating harmony: the function $InitMultisets$ initializes the duplication multisets for each harmony block, and the function $AnalyzeFiguringAndOuterVoices$ performs the analysis and updates the duplication multisets accordingly. For more details on the workings of these two functions, see Section 5.4.1.

Checking the inner voices is then done in two main steps:

1. Checking correspondence between the inner voices and the duplication multisets,
2. Checking voice leading rules for the inner voices.

In the first step we check, for each harmony block i in exercise e, if the pitches in the inner voices correspond to some element in $e.posDup[i].dup$. In case a duplication has been used that does not occur in $e.posFig[i].dup$, an error message is added to the log, saying that this duplication is not allowed. If a duplication has been used that is not the first element of $e.posDup[i].dup$, a warning message is added to the log, saying that this duplication is correct, but unlikely. The second step checks the harmony rules that concern all voices.

Again, a variable of type $Log[\,]$ is maintained, containing the error and warning messages.

6.4.2 Pseudocode

In this section we give pseudocode for the approach to checking the inner voices described above. Analyzing the outer voices and figuring, and checking the inner voices is done by function $AnalyzeAndCheckInnerVoices$, which takes the exercise as argument e, and the message log log of type $Log[\,]$. Precondition and postcondition of $log := AnalyzeAndCheckInnerVoices(e, log)$ for log of type $Log[\,]$ and e of type $Exercise$, are given by $P14$ and $P15'$ respectively. $P14$ and $P15$ are given on page 56. $AnalyzeAndCheckInnerVoices$ calls $InitMultisets$ and $AnalyzeFiguringAndOuterVoices$ (see Sections 5.4.3 and 5.4.4 respectively), and $CheckInnerVoices$, which performs the actual checking of the inner voices.

Pseudocode is given by:

func $AnalyzeAndCheckInnerVoices(e : Exercise; log : Log[\,]) : Log[\,]$
|[
  { $P14$ }
  $e := InitMultisets(e)$;
  { $P14 \wedge P11$ }
  $e := AnalyzeFiguringAndOuterVoices(e)$;
  { $P14 \wedge P12$ }
  $log := CheckInnerVoices(e, 0, log)$;
  { $P15$ }
  $AnalyzeAndCheckInnerVoices := log$
  { $P15'$ }
]|

$P15' : P15 \wedge AnalyzeAndCheckInnerVoices = log$

The function $CheckInnerVoices$ is specified in the next section.

6.4.3 CheckInnerVoices

This section specifies function $CheckInnerVoices$, introduced in the previous section. This function checks the inner voices and updates the message log accordingly. $CheckInnerVoices$ accepts an exercise e, a harmony block i, and a message log log as arguments, and returns an updated version of log. Precondition and postcondition of $log := CheckInnerVoices(e, 0, log)$ for e of type $Exercise$ and log of type $Log[\,]$, are given by $P14 \wedge P12$ and $P15$ respectively. $P14$ and $P15$ were defined on page 56. The postcondition of $log := CheckInnerVoices(e, i, log)$ for e of type $Exercise$, log of type $Log[\,]$ and i of type $\mathbb{N}$, is given by:

$$P19 : P14 \wedge \langle \forall j : i \leq j < e.numMeasures * \tfrac{e.meter.num}{e.harBlock} : LogInnerVoices.j.log \rangle$$

This predicate says that $P14$ holds (i.e. the figuring and outer voices were checked) and that for all harmony blocks from i onward, the inner voices were checked and the message log has been updated. Filling in $i = 0$ yields $P15$, the postcondition of $CheckInnerVoices(e, 0, log)$.

Following the approach explained in Section 6.4.1, pseudocode for $CheckInnerVoices$ is given by:

func $CheckInnerVoices(e : Exercise; i : \mathbb{N}; log : Log[\,]) : Log[\,]$
|[
  { $P12 \wedge P14$ }
  $log := CheckCorrespondenceInnerVoices(e, i, log)$;
  { $P12 \wedge LogCorrespondenceInnerVoices.i.log$ }
  $log := CheckVoiceLeadingRulesInnerVoices(e, i, log)$;
  { $P12 \wedge LogInnerVoices.i.log$ }
  if $(i+1) * \frac{e.harBlock}{e.meter.num} < e.numMeasures \rightarrow$
        $log := CheckInnerVoices(e, i+1, log)$
  [] $(i+1) * \frac{e.harBlock}{e.meter.num} = e.numMeasures \rightarrow$ skip
  fi;
  { $P19$ }
  $CheckOuterVoices := log$
  { $P19'$ }
]|

$$P19' : P19 \wedge CheckInnerVoices = log$$

The function $CheckCorrespondenceInnerVoices$ checks whether the notes in the non-given outer voice correspond to the figuring symbols written (step 1), and $CheckVoiceLeadingRulesInnerVoices$ checks the voice leading rules for the inner voices (step 2) These functions are specified in the next two sections.

CheckCorrespondenceInnerVoices

The function $CheckCorrespondenceInnerVoices(e : Exercise; i : \mathbb{N}; log : Log[\,])$ checks whether the notes for the inner voices filled in at harmony block i correspond to a legal duplication of chord tones for this harmony block. To this end, it is checked, for all notes in all voices that sound during harmony block i, if the combination of this note with the other notes in the other voices sounding simultaneously corresponds to a duplication of chord tones that can be found in $e.posDup[i].dup$. If a duplication has been used that does not occur in $e.posDup[i].dup$, an error message is added to log. If a duplication has been used that is not favored (i.e. it is in $e.posDup[i].dup$, but not the first entry), a warning message is added to log.

$CheckCorrespondenceInnerVoices$ accepts an exercise e, a harmony block i and a message log log as arguments, and returns an updated version of log. Precondition and postcondition of $log := CheckCorrespondenceInnerVoices(e, i, log)$ for log of type $Log[\,]$, e of type $Exercise$, and i of type $\mathbb{N}$, are given by $P12 \wedge P14$ and $P12 \wedge LogCorrespondenceInnerVoices.i.log$ respectively. $P12$ is defined in Section 5.4.4 and $P14$ is given on page 56. $LogCorrespondenceInnerVoices$ is given by:

$$
\begin{array}{l}
LogCorrespondenceInnerVoices.i.log : \\
\big\langle \forall v : 0 \le v < e.numVoices : \\
\quad \big\langle \forall n : 0 \le n < e.voices[v].numNotes \wedge \\
\qquad i * e.harBlock \le e.voices[v].notes[n].ontime < (i+1) * e.harBlock \vee \\
\qquad (e.voices[v].notes[n].ontime < i * e.harBlock \wedge \\
\qquad\quad e.voices[v].notes[n].ontime + e.voices[v].notes[n].duration > i * e.harBlock) \\
\qquad : \\
\qquad Q10.i.v.n \equiv \big\langle \exists j : 0 \le j \wedge log[j] = [\!| \, error; i * e.harBlock; \text{“note ”} \triangleright \\
\qquad\qquad\qquad\qquad e.voices[v].notes[n].pitch \triangleright \text{“ in voice ”} \triangleright v \triangleright \\
\qquad\qquad\qquad\qquad \text{“ is not a chord tone of ”} \triangleright e.figSymSeq[i] \, |\!] \big\rangle \\
\qquad \wedge \\
\qquad Q11.i.v.n \equiv \big\langle \exists j : 0 \le j \wedge log[j] = [\!| \, error; i * e.harBlock; \\
\qquad\qquad\qquad\qquad \text{“illegal duplication of chord tones”} \, |\!] \big\rangle \\
\qquad \wedge \\
\qquad Q12.i.v.n \equiv \big\langle \exists j : 0 \le j \wedge log[j] = [\!| \, warning; i * e.harBlock; \\
\qquad\qquad\qquad\qquad \text{“unfavored duplication of chord tones”} \, |\!] \big\rangle \\
\quad \big\rangle \\
\big\rangle
\end{array}
$$

This predicate says that, for all notes n in all voices that sound during harmony block i, the predicates $Q10..Q12$ hold, and updates log accordingly. $Q10$, defined on page 52, evaluates to *true* iff note n in voice v is not a chord tone of the figuring symbol in harmony block i. $Q11$, defined on page 89, evaluates to *true* iff a duplication of chord tones has been chosen that does not occur in $e.posDup[i].dup$. $Q12$ evaluates to *true* iff a duplication has been chosen that is in $e.posDup[i].dup$, but not as the first element (i.e. it is an allowed duplication, but not the preferred one). To this end, it is checked if the multiset of chord notes of this note, and the other three notes in the other voices sounding simultaneously, is an element of $\{e.posDup[i].dup[1], .., e.posDup[i].dup[e.numPos - 1]\}$ (i.e. an allowed duplication, but not the preferred one) or not. In order to find the notes that sound simultaneously with note n of voice v, we use, just like in $Q11$, the function $GetNoteAtTime$, using $e.voices[v].notes[n].ontime \uparrow i * e.harBlock$ as second argument; this is needed in case the ontime of note n comes before harmony block i. Like $Q11$, $Q12$ assumes exercise e to contain four voices:

$$
\begin{array}{l}
Q12.i.v.n : \\
\big[\\
\quad ChordNote(\\
\qquad e.voices[v].notes[n].pitch.tcd, \\
\qquad e.voices[v].notes[n].pitch.acc, \\
\qquad e.figSymSeq[i] \\
\quad), \\
\quad ChordNote(\\
\qquad GetNoteAtTime(e, e.voices[v].notes[n].ontime \uparrow i * e.harBlock, \\
\qquad\quad v + 1 \bmod e.numVoices).pitch.tcd, \\
\qquad GetNoteAtTime(e, e.voices[v].notes[n].ontime \uparrow i * e.harBlock, \\
\qquad\quad v + 1 \bmod e.numVoices).pitch.acc, \\
\qquad e.figSymSeq[i] \\
\quad), \\
\quad ChordNote(\\
\qquad GetNoteAtTime(e, e.voices[v].notes[n].ontime \uparrow i * e.harBlock, \\
\qquad\quad v + 2 \bmod e.numVoices).pitch.tcd, \\
\qquad GetNoteAtTime(e, e.voices[v].notes[n].ontime \uparrow i * e.harBlock, \\
\qquad\quad v + 2 \bmod e.numVoices).pitch.acc,
\end{array}
$$

$$
\begin{array}{l}
\quad e.figSymSeq[i] \\
\quad), \\
\quad ChordNote(\\
\quad\quad GetNoteAtTime(e, e.voices[v].notes[n].ontime \uparrow i * e.harBlock, \\
\quad\quad\quad v + 3 \bmod e.numVoices).pitch.tcd, \\
\quad\quad GetNoteAtTime(e, e.voices[v].notes[n].ontime \uparrow i * e.harBlock, \\
\quad\quad\quad v + 3 \bmod e.numVoices).pitch.acc, \\
\quad\quad e.figSymSeq[i] \\
\quad) \\
] \in \{e.posDup[i].dup[1], .., e.posDup[i].dup[e.numPos - 1]\}
\end{array}
$$

CheckVoiceLeadingRulesInnerVoices

The function $CheckVoiceLeadingRulesAllVoices(e : Exercise; i : \mathbb{N}; log : Log[\,])$ checks whether the inner voices $e.voices[e.numVoices - j].notes$, $2 \leq j < e.numVoices$ are correct with respect to the harmony rules that concern the voice leading of the inner voices. Error and warning messages are added to *log*.

$CheckVoiceLeadingRulesInnerVoices$ accepts an exercise e, a harmony block i and a message log *log* as arguments, and returns an updated version of *log*. Precondition and postcondition of $log := CheckVoiceLeadingRulesInnerVoices(e, i, log)$ for *log* of type $Log[\,]$, e of type $Exercise$ and i of type $\mathbb{N}$, are given by $P12 \wedge LogCorrespondenceInnerVoices.i.log$ and $P12 \wedge LogInnerVoices.i.log$ respectively. $LogCorrespondenceInnerVoices$ is given in the previous section, $P12$ is defined in Section 5.4.4. $LogInnerVoices$ is given by:

$$
\begin{array}{l}
LogInnerVoices.i.log : \\
LogCorrespondenceInnerVoices.i.log \wedge LogVoiceLeadingRulesInnerVoices.i.log
\end{array}
$$

$$
\begin{array}{l}
LogVoiceLeadingRulesInnerVoices.i.log : \\
LogRuleC1P0R0.i.log \wedge LogRuleC1P0R1.i.log \wedge LogRuleC1P0R2.i.log \wedge \\
LogRuleC1P0R6.i.log \wedge LogRuleC1P0R7.i.log \wedge \\
(e.section > 0 \Rightarrow \\
\quad LogRuleC1P1R2.i.log \wedge LogRuleC1P1R3.i.log \wedge LogRuleC1P1R4.i.log \wedge \\
\quad LogRuleC1P1R5.i.log) \wedge \\
(e.section > 1 \Rightarrow LogRuleC1P3R4.i.log)
\end{array}
$$

Every rule has an associated predicate that maintains the message log. Rule C1P1R2 concerns all voices, the suffix SB in the rule name indicates that here only the outer voices are checked. The predicates for rules C1P0R8 and C1P1R2SB are given in Appendix A.3.2.

Every rule has an associated predicate that maintains the message log. The predicate for rule C1P3R4 is given in Appendix A.3.3.

Chapter 7

Requirements evaluation

This chapter evaluates the requirements, given in Section 2.1, on an eventual didactic software system in which the algorithm presented in Chapters 5 and 6 of this thesis would be incorporated. The requirements on the algorithm itself, given in Section 2.2, can easily be seen to be met. The system requirements deserve some further commentary.

In the next sections, we consider these system requirements in light of the algorithm presented in Chapters 5 and 6. We discuss to which extent the algorithm would permit the development of a system described by these requirements. There are three requirement categories: general requirements, requirements with respect to checking exercises, and requirements with respect to generating hints for partially completed exercises. These three categories are discussed in the next three sections.

7.1 General requirements

We remark the following regarding the two general requirements:

- [G0] As a result of an extensive implementation analysis we have conducted, we can confidently say that the algorithm presented in this thesis can in its entirety be programmed as a plug-in to the music notation software system Finale 2006.
- [G1] The algorithm is capable of working with arbitrary harmony rule sets. We have explicitly shown this for the first three sections of [dCM], but the extendability towards other rule sets is clear as well.

7.2 Checking an exercise

The requirements for checking exercises are discussed below:

- [C0] The algorithm checks harmony in three distinct phases: first, it checks the figuring (covered by Section 6.2), then it checks the outer voices (Section 6.3), and finally it checks the inner voices (Section 6.4). This checking is done with the rules from that section of [dCM] from which the algorithm is taken.
- [C1] The algorithm is set up in such a way that it checks either
  - the figuring of the given voice,
  - the figuring and the outer voices, or

  - figuring, outer voices and inner voices.

- [C2] The algorithm assumes that the exercise to be checked has been filled in completely.
- [C3] We can remark the following with respect to the error messages:
  - the exercise is correct if and only if the log file returned by the algorithm is empty,
  - for each error, there is an entry in the log file with
    * a textual message that describes the nature of the error, and
    * the position at which the error has been found.
- [C4] For each error, the log file returned by the algorithm contains an entry that says at which position the error has been found.
- [C5] The algorithm can work with arbitrary rule sets to check harmony.
- [C6] The algorithm permits two types of rules, namely strict and soft rules, and each entry in the log file contains a field that indicates whether the message is an error message or a warning message.
- [C7 and C8] Since all log entries contain the positions at which the errors have been found, it is possible to arrange the error messages in any way necessary.

7.3 Giving a hint

The following can be observed with respect to the system requirements for giving hints:

- [H0] The algorithm for generating harmony follows the same three steps (construction of the figuring, the non-given outer voice, and the inner voices, in this order) as those the student needs to follow when completing an exercise using the software system.
- [H1] For each step taken by the system in order to give a hint, we remark the following:
  - it is possible to use (part of) the algorithm for checking harmony to check some incomplete input,
  - the algorithm for checking harmony can also be used to give feedback on errors made in this input,
  - the algorithm for generating harmony can be used to complete the exercise starting from the material given by the student,
  - the hint to be given can be based on the output of the algorithm for generating harmony,
  - the algorithm for generating harmony can again be used to complete the exercise starting with part of the input harmony.
- [H2] The algorithm for generating harmony can be used to complete the input harmony; the algorithm's output can then be used to construct a proper hint.
- [H3] Since the algorithm can work with the harmony rules from [dCM], its output (and therefore also the hint constructed from this output) will obey these rules.

As the above observations show, the algorithm is constructed in such a way that it can be used in a didactic software tool that checks some complete input harmony and gives hints for some incomplete input harmony. The general requirements and the requirements for checking harmony are met very closely. In order to generate hints, some method would have to be devised to construct useful hints from the algorithm's output, but also here the algorithm fits the system requirements well.

Chapter 8

Conclusions

This chapter gives an evaluation of the work presented in this thesis. Section 8.1 gives a general evaluation of the course of the project, Section 8.2 discusses the main accomplishments that have been achieved, and Section 8.3 gives recommendations for continuing the work presented in this thesis and for implementing the algorithm.

8.1 General evaluation of the project

Before actually being able to start working on a sensible harmony algorithm, it took a lot of effort to determine what it actually was that this algorithm was supposed to do. Getting the specific aims of the project on paper turned out to be a significant part of the project itself. The results of this work are given in Section 2.

Apart from this, a time consuming implementation analysis was needed in order to determine possibilities of an eventual realization of the algorithm in a software system. The music notation software system Finale 2006, with its alleged potential when it comes to programming plug-ins, and its ease-of-use and intuitive user interface, soon appeared to be by far the preferred implementation environment. However, documentation on how to program plug-ins turned out to be very poor, and it took a lot of time and effort to determine how the implementation environment would restrict, or stimulate thinking of the development of our algorithm.

Also, we conducted a domain analysis to reveal the didactic process in harmony courses at the Fontys Conservatory of Tilburg. The results of this analysis are given in Section 1.2.

Having done these preliminary studies, most of the project time was invested in the construction of the algorithm for generating and checking tonal harmony presented in Chapters 5 and 6. The major difficulty here turned out to be that there is no consensus regarding the harmony rules, i.e. there is not one definitive harmony rule set. Some harmony rules, as presented in [dCM], are rigorous and unquestionable, many of them, however, are open to discussion and interpretation, if not ambiguous or over-general. Also, many rules should be understood as heuristics rather than actual rules. The project did not aim for the construction of a general, formal set of harmony rules, but for the construction of an algorithm that, given such a rule set, can check harmony exercises and generate solutions. Keeping these two goals separated turned out to be a key difficulty.

8.2 Main accomplishments

The main accomplishments of the algorithm design presented in this thesis are:

- The algorithm is set up in such a way that it corresponds to the requirements on an eventual didactic software system for harmony courses. These requirements are given in Section 2.1. In Chapter 7 we have discussed this correspondence in detail.
- The algorithm successfully accommodates all rules in [dCM] that we have studied (i.e. the rules of the introduction and the first three sections of [dCM]), and we expect that also other rule sets can be incorporated into the algorithm. Extendability towards other harmony rules has therefore been covered as well.
- All difficult subproblems that arise when filling in a harmony exercise (e.g. analysis of the given voice, melody writing, voice leading, chord tone duplication) are addressed by the algorithm. Moreover, these subproblems are separated and dealt with in isolation, resulting in the fact that the algorithm does not need to consider several difficult issues at once when making its decisions.
- The fact that the rules of harmony are often not rigorous, is reflected by the use of two rule categories, namely strict and soft rules, and of probability values denoting the likelihood with which certain notes and figuring symbols are chosen.
- The algorithm works in such a way that human behavior is closely simulated. This makes the inner workings of the algorithm very transparent and intuitive.
- The three steps taken to generate harmony (generate figuring, non-given outer voice, inner voices) are closely related to the three steps taken to check harmony (check figuring, outer voices, inner voices), which results in the fact that much of the effort invested in one problem leads to a significant simplification of the other.
- The approaches to generating figuring, generating the non-given outer voice and generating inner voices are of the same shape, which makes it manageable to keep a complete overview of the algorithm. The same goes for the three steps in checking harmony.

8.3 Directions for future work

This section indicates some possibilities for continuing the work done in this project. Section 8.3.1 discusses some issues that deserve further researching, and Section 8.3.2 explains which steps need to be followed to implement the algorithm.

8.3.1 Open research issues

The following points can be considered in order to obtain a better understanding of some intricate problems we have come across, and possibly to obtain better results in the output of the algorithm.

The algorithm we have presented relies on meta-data that is present in each exercise. This was not only a necessity, but for some cases also turned out to be quite natural: each exercise is put together by a harmony teacher with a specific purpose, and it seems contrived not to include this purpose in the exercise in one way or another as meta-data. The necessity lies in the fact that a computation of this meta-data would cost so much effort that it had to fall out of the time scope of this project. However, one could strive for the elimination of (some of) this meta-data by devising a mechanism that computes it autonomically. To make this possible, separate functions are available into which these operations can be implemented.

When checking harmony, we have proposed to output warning messages whenever a figuring symbol (when checking the figuring) or a note (when checking the non-given outer voice) has been written that has a low probability of occurrence. In order to determine when a probability is low enough to yield a warning message, we compute a probability bound from the some constant values,

which need to result from statistical analysis and evaluation. We can however not guarantee that a unique value for these constants exists which works satisfactorily in any case. We have therefore chosen to create two functions, $FigProbBound$ and $NGOVProbBound$ respectively (see page 60 and 63), that can be reimplemented to give room to a more sophisticated way of determining these bounds.

Another issue that is worth further discussion is the determination of relatively weak and strong beats. The function *Stress* that we have proposed on page 88 uses a technique based on observations by F. Lerdahl and R.S. Jackendoff in [LJ83]. These observations are however far from undebatable and could in some cases lead to undesirable effects in the algorithm's output. These effects can be restricted by relaxing the harmony rules concerning relatively weak and strong beats, but it would of course be best to aim for a better formal understanding of the notion of stress level.

In our study of grammar-based approaches to harmony generation, we have constructed a regular grammar for the language of permitted chord progressions with main triads in all inversions (covered by the first three sections of [dCM]). This grammar is given in Appendix C. It is possible to replace a large number of figuring rules by such a grammar: instead of having to check each figuring rule separately, the grammar can be used to find out if a certain figuring symbol, or sequence of figuring symbols, is permitted or not. Considering the performance of the algorithm, this may be a point of interest. The use of such a grammar would obviously lead to a decrease in the number of rules that need to be checked and thus to more transparency in the harmony rule set.

8.3.2 Implementing the algorithm

The following steps need to be taken to implement the algorithm presented in Chapters 5 and 6 of this thesis.

Before being able to implement the algorithm, all rules given in the intended sections of [dCM] need to be formalized and put in terms of predicate logic. The rules of the introduction and the first three sections of [dCM] are listed in Appendix B.

For an implementation of the algorithm presented in Chapters 5 and 6, we recommend writing a plug-in to the music notation system Finale 2006. Plug-ins need to be programmed in C++. As mentioned, documentation on how to program plug-ins for Finale is sparse, but there is no doubt as to whether the algorithm can be implemented in its entirety as a Finale plug-in. Also, there is no need to worry about an extensive and user-friendly user interface, this is provided by Finale.

Prior to using the implemented software system, a statistical analysis must be performed to determine probabilities for chord symbols and notes in the outer voices. To this end, we recommend analyzing a large number of correctly filled in harmony exercises, which need to be obtained from the harmony teachers of the Fontys Conservatory.

Having implemented the algorithm and having performed a statistical analysis of existing harmony exercises, some evaluation cycles will have to be carried out in order to determine the constant probability bounds used by the algorithm. In this stage, also the harmony rule set will have to be refined. It is expected that the rule set taken from [dCM] will, due to certain rules being ambiguous or too general, disapprove of certain correct solutions and allow for certain incorrect solutions. Only sufficient testing and meticulous precision when formalizing the rules will result in a rule set that works satisfactorily.

When the above steps have been followed and the algorithm functions adequately, one could strive for eliminating some of the meta-data. We expect that every piece of meta-data is worth a study on its own. When considering full-fledged automatic harmonization, there is still a long way to go.

Appendix A

Algorithm details

This appendix contains a detailed discription of some of the aspects of the algorithm introduced in Chapters 4 up to 6. Section A.1 concerns the specification primitives presented in Chapter 4, Section A.2 concerns the algorithm for checking harmony covered by Chapter 5, and Section A.3 relates to the algorithm for checking harmony discussed in Chapter 6.

A.1 Specification primitives

This section contains the details concerning Chapter 4, Specification primitives.

A figuring symbol is a variable of type $FigSym$. This type is given by:

type $FigSym = S_0 \cup S_1 \cup S_2 \cup S_3 \cup S_4 \cup S_5 \cup S_6 \cup S_7 \cup \perp$

The figuring symbols are contained by the sets $S_0..S_7$. S_0 contains the figuring symbols of triads in all inversions:

$S_0 =$
$\{\mathrm{I},\mathrm{I}^6,\mathrm{I}^6_4,\mathrm{I}_{pic},\mathrm{II},\mathrm{II}^6,\mathrm{II}^6_4,\mathrm{II}_{md},\mathrm{II}^6_{md},\mathrm{II}^6_{4md},\mathrm{II}_{dm},\mathrm{II}^6_{dm},\mathrm{II}^6_{4dm},\mathrm{III},\mathrm{III}^6,\mathrm{III}^6_4,\mathrm{III}_{aeol},\mathrm{III}^6_{aeol},$
$\mathrm{III}^6_{4aeol},\mathrm{IV},\mathrm{IV}^6,\mathrm{IV}^6_4,\mathrm{IV}_{md},\mathrm{IV}^6_{md},\mathrm{IV}^6_{4md},\mathrm{IV}_{dm},\mathrm{IV}^6_{dm},\mathrm{IV}^6_{4dm},\mathrm{IV}^{\emptyset},\mathrm{V},\mathrm{V}^6,\mathrm{V}^6_4,\mathrm{V}_o,\mathrm{V}^6_o,$
$\mathrm{VI},\mathrm{VI}^6,\mathrm{VI}^6_4,\mathrm{VI}_{md},\mathrm{VI}^6_{md},\mathrm{VI}^6_{4md},\mathrm{VII},\mathrm{VII}^6,\mathrm{VII}^6_4,\mathrm{VII}_{aeol},\mathrm{VII}^6_{aeol},\mathrm{VII}^6_{4aeol}\}$

S_1 contains the figuring symbols of seventh chords in all inversions:

$S_1 =$
$\{\mathrm{I}^7,\mathrm{I}^6_5,\mathrm{I}^4_3,\mathrm{I}_2,\mathrm{II}^7,\mathrm{II}^6_5,\mathrm{II}^4_3,\mathrm{II}_2,\mathrm{II}^4_3,\mathrm{II}^7_{md},\mathrm{II}^6_{5md},\mathrm{II}^4_{3md},\mathrm{II}_{2md},\mathrm{II}^7_{dm},\mathrm{II}^6_{5dm},\mathrm{II}^4_{3dm},\mathrm{II}_{2dm},\mathrm{III}^7,$
$\mathrm{III}^6_5,\mathrm{III}^4_3,\mathrm{III}_2,\mathrm{III}^7_{aeol},\mathrm{III}^6_{5aeol},\mathrm{III}^4_{3aeol},\mathrm{III}_{2aeol},\mathrm{IV}^7,\mathrm{IV}^6_5,\mathrm{IV}^4_3,\mathrm{IV}_2,\mathrm{IV}^7_{md},\mathrm{IV}^6_{5md},\mathrm{IV}^4_{3md},$
$\mathrm{IV}_{2md},\mathrm{IV}^7_{dm},\mathrm{IV}^6_{5dm},\mathrm{IV}^4_{3dm},\mathrm{IV}_{2dm},\mathrm{IV}^{\emptyset}_5,\mathrm{V}^7,\mathrm{V}^6_5,\mathrm{V}^4_3,\mathrm{V}_2,\mathrm{VI}^7,\mathrm{VI}^6_5,\mathrm{VI}^4_3,\mathrm{VI}_2,\mathrm{VI}^7_{md},$
$\mathrm{VI}^6_{5md},\mathrm{VI}^4_{3md},\mathrm{VI}_{2md},\mathrm{VII}^7,\mathrm{VII}^6_5,\mathrm{VII}^4_3,\mathrm{VII}_2,\mathrm{VII}^7_{aeol},\mathrm{VII}^6_{5aeol},\mathrm{VII}^4_{3aeol},\mathrm{VII}_{2aeol}\}$

The sets S_2 and S_3 contain the figuring symbols of chords that start a group of secondary (sub)dominants. S_2 contains triads, S_3 contains seventh chords:

$S_2 =$

$\{(\mathrm{I},(\mathrm{I}^6,(\mathrm{I}^6_4,(\mathrm{II},(\mathrm{II}^6,(\mathrm{II}^6_4,(\mathrm{II}_{md},(\mathrm{II}^6_{md},(\mathrm{II}^6_{4md},(\mathrm{II}_{dm},(\mathrm{II}^6_{dm},(\mathrm{II}^6_{4dm},(\mathrm{III},(\mathrm{III}^6,(\mathrm{III}^6_4,$
$(\mathrm{III}_{aeol},(\mathrm{III}^6_{aeol},(\mathrm{III}^6_{4aeol},(\mathrm{IV},(\mathrm{IV}^6,(\mathrm{IV}^6_4,(\mathrm{IV}_{md},(\mathrm{IV}^6_{md},(\mathrm{IV}^6_{4md},(\mathrm{IV}_{dm},(\mathrm{IV}^6_{dm},$
$(\mathrm{IV}^6_{4dm},(\mathrm{IV}^\emptyset,(\mathrm{V},(\mathrm{V}^6,(\mathrm{V}^6_4,(\mathrm{V}_o,(\mathrm{V}^6_o,(\mathrm{VI},(\mathrm{VI}^6,(\mathrm{VI}^6_4,(\mathrm{VI}_{md},(\mathrm{VI}^6_{md},(\mathrm{VI}^6_{4md},(\mathrm{VII},$
$(\mathrm{VII}^6,(\mathrm{VII}^6_4,(\mathrm{VII}_{aeol},(\mathrm{VII}^6_{aeol},(\mathrm{VII}^6_{4aeol}\}$

$S_3 =$

$\{(\mathrm{I}^7,(\mathrm{I}^6_5,(\mathrm{I}^4_3,(\mathrm{I}_2,(\mathrm{II}^7,(\mathrm{II}^6_5,(\mathrm{II}^4_3,(\mathrm{II}_2,(\mathrm{II}^4_3,(\mathrm{II}^7_{md},(\mathrm{II}^6_{5md},(\mathrm{II}^4_{3md},(\mathrm{II}_{2md},(\mathrm{II}^7_{dm},(\mathrm{II}^6_{5dm},$
$(\mathrm{II}^4_{3dm},(\mathrm{II}_{2dm},(\mathrm{III}^7,(\mathrm{III}^6_5,(\mathrm{III}^4_3,(\mathrm{III}_2,(\mathrm{III}^7_{aeol},(\mathrm{III}^6_{5aeol},(\mathrm{III}^4_{3aeol},(\mathrm{III}_{2aeol},(\mathrm{IV}^7,(\mathrm{IV}^6_5,$
$(\mathrm{IV}^4_3,(\mathrm{IV}_2,(\mathrm{IV}^7_{md},(\mathrm{IV}^6_{5md},(\mathrm{IV}^4_{3md},(\mathrm{IV}_{2md},(\mathrm{IV}^7_{dm},(\mathrm{IV}^6_{5dm},(\mathrm{IV}^4_{3dm},(\mathrm{IV}_{2dm},(\mathrm{IV}^\emptyset_5,(\mathrm{V}^7,$
$(\mathrm{V}^6_5,(\mathrm{V}^4_3,(\mathrm{V}_2,(\mathrm{VI}^7,(\mathrm{VI}^6_5,(\mathrm{VI}^4_3,(\mathrm{VI}_2,(\mathrm{VI}^7_{md},(\mathrm{VI}^6_{5md},(\mathrm{VI}^4_{3md},(\mathrm{VI}_{2md},(\mathrm{VII}^7,(\mathrm{VII}^6_5,(\mathrm{VII}^4_3,$
$(\mathrm{VII}_2,(\mathrm{VII}^7_{aeol},(\mathrm{VII}^6_{5aeol},(\mathrm{VII}^4_{3aeol},(\mathrm{VII}_{2aeol}\}$

The sets S_4 and S_5 contain the figuring symbols of chords that end a group of secondary (sub)dominants:

$S_4 =$

$\{(\mathrm{I}),\mathrm{I}^6),\mathrm{I}^6_4),\mathrm{II}),\mathrm{II}^6),\mathrm{II}^6_4),\mathrm{II}_{md}),\mathrm{II}^6_{md}),\mathrm{II}^6_{4md}),\mathrm{II}_{dm}),\mathrm{II}^6_{dm}),\mathrm{II}^6_{4dm}),\mathrm{III}),\mathrm{III}^6),\mathrm{III}^6_4),\mathrm{III}_{aeol}),$
$\mathrm{III}^6_{aeol}),\mathrm{III}^6_{4aeol}),\mathrm{IV}),\mathrm{IV}^6),\mathrm{IV}^6_4),\mathrm{IV}_{md}),\mathrm{IV}^6_{md}),\mathrm{IV}^6_{4md}),\mathrm{IV}_{dm}),\mathrm{IV}^6_{dm}),\mathrm{IV}^6_{4dm}),\mathrm{IV}^\emptyset),\mathrm{V}),$
$\mathrm{V}^6),\mathrm{V}^6_4),\mathrm{V}_o),\mathrm{V}^6_o),\mathrm{VI}),\mathrm{VI}^6),\mathrm{VI}^6_4),\mathrm{VI}_{md}),\mathrm{VI}^6_{md}),\mathrm{VI}^6_{4md}),\mathrm{VII}),\mathrm{VII}^6),\mathrm{VII}^6_4),\mathrm{VII}_{aeol}),$
$\mathrm{VII}^6_{aeol}),\mathrm{VII}^6_{4aeol})\}$

$S_5 =$

$\{\mathrm{I}^7),\mathrm{I}^6_5),\mathrm{I}^4_3),\mathrm{I}_2),\mathrm{II}^7),\mathrm{II}^6_5),\mathrm{II}^4_3),\mathrm{II}_2),\mathrm{II}^4_3),\mathrm{II}^7_{md}),\mathrm{II}^6_{5md}),\mathrm{II}^4_{3md}),\mathrm{II}_{2md}),\mathrm{II}^7_{dm}),\mathrm{II}^6_{5dm}),$
$\mathrm{II}^4_{3dm}),\mathrm{II}_{2dm}),\mathrm{III}^7),\mathrm{III}^6_5),\mathrm{III}^4_3),\mathrm{III}_2),\mathrm{III}^7_{aeol}),\mathrm{III}^6_{5aeol}),\mathrm{III}^4_{3aeol}),\mathrm{III}_{2aeol}),\mathrm{IV}^7),\mathrm{IV}^6_5),$
$\mathrm{IV}^4_3),\mathrm{IV}_2),\mathrm{IV}^7_{md}),\mathrm{IV}^6_{5md}),\mathrm{IV}^4_{3md}),\mathrm{IV}_{2md}),\mathrm{IV}^7_{dm}),\mathrm{IV}^6_{5dm}),\mathrm{IV}^4_{3dm}),\mathrm{IV}_{2dm}),\mathrm{IV}^\emptyset_5),\mathrm{V}^7),$
$\mathrm{V}^6_5),\mathrm{V}^4_3),\mathrm{V}_2),\mathrm{VI}^7),\mathrm{VI}^6_5),\mathrm{VI}^4_3),\mathrm{VI}_2),\mathrm{VI}^7_{md}),\mathrm{VI}^6_{5md}),\mathrm{VI}^4_{3md}),\mathrm{VI}_{2md}),\mathrm{VII}^7),\mathrm{VII}^6_5),\mathrm{VII}^4_3),$
$\mathrm{VII}_2),\mathrm{VII}^7_{aeol}),\mathrm{VII}^6_{5aeol}),\mathrm{VII}^4_{3aeol}),\mathrm{VII}_{2aeol})\}$

The sets S_6 and S_7 contain the figuring symbols of chords that start and end a group of secondary (sub)dominants:

$S_6 =$

$\{(\mathrm{I}),(\mathrm{I}^6),(\mathrm{I}^6_4),(\mathrm{II}),(\mathrm{II}^6),(\mathrm{II}^6_4),(\mathrm{II}_{md}),(\mathrm{II}^6_{md}),(\mathrm{II}^6_{4md}),(\mathrm{II}_{dm}),(\mathrm{II}^6_{dm}),(\mathrm{II}^6_{4dm}),(\mathrm{III}),$
$(\mathrm{III}^6),(\mathrm{III}^6_4),(\mathrm{III}_{aeol}),(\mathrm{III}^6_{aeol}),(\mathrm{III}^6_{4aeol}),(\mathrm{IV}),(\mathrm{IV}^6),(\mathrm{IV}^6_4),(\mathrm{IV}_{md}),(\mathrm{IV}^6_{md}),(\mathrm{IV}^6_{4md}),$
$(\mathrm{IV}_{dm}),(\mathrm{IV}^6_{dm}),(\mathrm{IV}^6_{4dm}),(\mathrm{IV}^\emptyset),(\mathrm{V}),(\mathrm{V}^6),(\mathrm{V}^6_4),(\mathrm{V}_o),(\mathrm{V}^6_o),(\mathrm{VI}),(\mathrm{VI}^6),(\mathrm{VI}^6_4),(\mathrm{VI}_{md}),$
$(\mathrm{VI}^6_{md}),(\mathrm{VI}^6_{4md}),(\mathrm{VII}),(\mathrm{VII}^6),(\mathrm{VII}^6_4),(\mathrm{VII}_{aeol}),(\mathrm{VII}^6_{aeol}),(\mathrm{VII}^6_{4aeol})\}$

$S_7 =$

$\{(\mathrm{I}^7),(\mathrm{I}^6_5),(\mathrm{I}^4_3),(\mathrm{I}_2),(\mathrm{II}^7),(\mathrm{II}^6_5),(\mathrm{II}^4_3),(\mathrm{II}_2),(\mathrm{II}^4_3),(\mathrm{II}^7_{md}),(\mathrm{II}^6_{5md}),(\mathrm{II}^4_{3md}),(\mathrm{II}_{2md}),$
$(\mathrm{II}^7_{dm}),(\mathrm{II}^6_{5dm}),(\mathrm{II}^4_{3dm}),(\mathrm{II}_{2dm}),(\mathrm{III}^7),(\mathrm{III}^6_5),(\mathrm{III}^4_3),(\mathrm{III}_2),(\mathrm{III}^7_{aeol}),(\mathrm{III}^6_{5aeol}),$
$(\mathrm{III}^4_{3aeol}),(\mathrm{III}_{2aeol}),(\mathrm{IV}^7),(\mathrm{IV}^6_5),(\mathrm{IV}^4_3),(\mathrm{IV}_2),(\mathrm{IV}^7_{md}),(\mathrm{IV}^6_{5md}),(\mathrm{IV}^4_{3md}),(\mathrm{IV}_{2md}),$
$(\mathrm{IV}^7_{dm}),(\mathrm{IV}^6_{5dm}),(\mathrm{IV}^4_{3dm}),(\mathrm{IV}_{2dm}),(\mathrm{IV}^\emptyset_5),(\mathrm{V}^7),(\mathrm{V}^6_5),(\mathrm{V}^4_3),(\mathrm{V}_2),(\mathrm{VI}^7),(\mathrm{VI}^6_5),(\mathrm{VI}^4_3),$
$(\mathrm{VI}_2),(\mathrm{VI}^7_{md}),(\mathrm{VI}^6_{5md}),(\mathrm{VI}^4_{3md}),(\mathrm{VI}_{2md}),(\mathrm{VII}^7),(\mathrm{VII}^6_5),(\mathrm{VII}^4_3),(\mathrm{VII}_2),(\mathrm{VII}^7_{aeol}),$
$(\mathrm{VII}^6_{5aeol}),(\mathrm{VII}^4_{3aeol}),(\mathrm{VII}_{2aeol})\}$

A.2 Generating harmony

This section contains the details concerning Chapter 5, Generating harmony.

A.2.1 Analysis and figuring of given voice

Tables A.1, A.2, and A.3 give the figuring tables for bass notes in the major mode, soprano notes in the minor mode, and bass notes in the minor mode, respectively.

The predicates $Q3..5$ are defined as follows:

$$
\begin{array}{l}
Q3.i: \\
e.posFig[i].numPos = \\
\quad |\, \langle \cap n : 0 \le n < e.voices[e.givenVoice].numNotes \wedge \\
\qquad i * e.harBlock \le e.voices[e.givenVoice].notes[n].ontime < (i+1) * e.harBlock \wedge \\
\qquad n \notin \langle \cup m : 0 \le m < |\, e.ornamentalNotes \,| : e.ornamentalNotes[m].ornNoteNumber \rangle : \\
\qquad\quad FigTable(e, n) \\
\quad \rangle \\
\quad \cap \\
\quad \langle \cap n, l : 0 < n, l < e.voices[e.givenVoice].numNotes \wedge \\
\qquad i * e.harBlock \le e.voices[e.givenVoice].notes[n].ontime < (i+1) * e.harBlock \wedge \\
\qquad n \in \langle \cup m : 0 \le m < |\, e.ornamentalNotes \,| : e.ornamentalNotes[m].ornNoteNumber \rangle \wedge \\
\qquad [\![\, n, l \,]\!] \in e.ornamentalNotes : \\
\qquad\quad FigTable(e, l) \\
\quad \rangle \,|
\end{array}
$$

$$
\begin{array}{l}
Q4.i: \\
e.posFig[i].fig = \\
\quad \langle \cap n : 0 \le n < e.voices[e.givenVoice].numNotes \wedge \\
\qquad i * e.harBlock \le e.voices[e.givenVoice].notes[n].ontime < (i+1) * e.harBlock \wedge \\
\qquad n \notin \langle \cup m : 0 \le m < |\, e.ornamentalNotes \,| : e.ornamentalNotes[m].ornNoteNumber \rangle : \\
\qquad\quad FigTable(e, n) \\
\quad \rangle \\
\quad \cap \\
\quad \langle \cap n, l : 0 < n, l < e.voices[e.givenVoice].numNotes \wedge \\
\qquad i * e.harBlock \le e.voices[e.givenVoice].notes[n].ontime < (i+1) * e.harBlock \wedge \\
\qquad n \in \langle \cup m : 0 \le m < |\, e.ornamentalNotes \,| : e.ornamentalNotes[m].ornNoteNumber \rangle \wedge \\
\qquad [\![\, n, l \,]\!] \in e.ornamentalNotes : \\
\qquad\quad FigTable(e, l) \\
\quad \rangle
\end{array}
$$

$$
\begin{array}{l}
Q5.i: \\
\langle \forall j : 0 \le j < e.posFig[i].numPos : e.posFig[i].prob[j] = \\
\quad \Big(\\
\quad \langle \prod n, h : 0 \le n < e.voices[e.givenVoice].numNotes \wedge \\
\qquad i * e.harBlock \le e.voices[e.givenVoice].notes[n].ontime < (i+1) * e.harBlock \wedge \\
\qquad n \notin \langle \cup m : 0 \le m < |\, e.ornamentalNotes \,| : e.ornamentalNotes[m].ornNoteNumber \rangle \wedge \\
\qquad 0 \le h < |FigTable(e, n)| \wedge FigTable(e, n)[h] = e.posFig[i].fig[j] : \\
\qquad\quad FigProbTable(e, n)[h] \\
\quad \rangle. \\
\quad \langle \prod n, l, h : 0 \le n, l < e.voices[e.givenVoice].numNotes \wedge \\
\qquad i * e.harBlock \le e.voices[e.givenVoice].notes[n].ontime < (i+1) * e.harBlock \wedge
\end{array}
$$

bass note pitch (tcd mod 7, acc)	possible figurings
(0,0)	I, IV_4^6, IV_{4md}^6
(1,0)	V_4^6
(2,0)	I^6
(3,0)	IV, IV_{md}
(4,0)	V, V_o, I_4^6
(5,-1)	IV_{md}^6, $\mathrm{IV}^{\emptyset}$
(5,0)	IV^6
(6,0)	V^6, V_o^6

Table A.1: Figuring table for bass notes, major mode, major triads in all inversions

soprano note pitch (tcd mod 7, acc)	possible figurings
(0,0)	I, I^6, I_4^6, I_{pic}, IV, IV^6, IV_4^6, $\mathrm{IV}^{\emptyset}$, IV_{dm}, IV_{dm}^6, IV_{4dm}^6
(1,0)	V, V^6, V_4^6, V_{aeol}^6
(1,1)	V_o, V_o^6
(2,0)	I, I_4^6
(2,1)	I_{pic}
(3,0)	IV^6, IV, IV_4^6, IV_{dm}, IV_{dm}^6, IV_{4dm}^6
(3,1)	$\mathrm{IV}^{\emptyset}$
(4,0)	V, V^6, V_4^6, V_o, V_o^6, V_{aeol}^6, I, I^6, I_4^6
(5,0)	IV, IV_4^6
(5,1)	IV_{dm}, IV_{4dm}^6
(6,0)	V, V_4^6

Table A.2: Figuring table for soprano notes, minor mode, major triads in all inversions

bass note pitch (tcd mod 7, acc)	possible figurings
(0,0)	I, IV_4^6, IV_{4dm}^6
(1,0)	V_4^6
(2,0)	I^6
(3,0)	IV, IV_{dm}
(4,0)	V, V_o, I_4^6
(5,0)	IV^6, $\mathrm{IV}^{\emptyset}$
(5,1)	IV_{dm}^6
(6,0)	V_{aeol}^6
(6,1)	V^6, V_o^6

Table A.3: Figuring table for bass notes, minor mode, major triads in all inversions

$$
\begin{array}{l}
\quad n \in \langle \cup m : 0 \le m < |\ e.ornamentalNotes\ | : e.ornamentalNotes[m].ornNoteNumber \rangle \wedge \\
\quad [\![\ n, l\ ]\!] \in e.ornamentalNotes \wedge \\
\quad 0 \le h < |FigTable(e,l)| \wedge FigTable(e,l)[h] = e.posFig[i].fig[j] : \\
\qquad FigProbTable(e,l)[h] \\
\rangle \\
) \\
/ \\
\langle \sum k : 0 \le k < e.posFig[i].numPos : \\
\quad (\\
\quad \langle \prod n, h : 0 \le n < e.voices[e.givenVoice].numNotes \wedge \\
\qquad i * e.harBlock \le e.voices[e.givenVoice].notes[n].ontime < (i+1) * e.harBlock \wedge \\
\qquad n \notin \langle \cup m : 0 \le m < |\ e.ornamentalNotes\ | : e.ornamentalNotes[m].ornNoteNumber \rangle \wedge \\
\qquad 0 \le h < |FigTable(e,n)| \wedge FigTable(e,n)[h] = e.posFig[i].fig[k] : \\
\qquad\quad FigProbTable(e,n)[h] \\
\quad \rangle . \\
\quad \langle \prod n, l, h : 0 \le n, l < e.voices[e.givenVoice].numNotes \wedge \\
\qquad i * e.harBlock \le e.voices[e.givenVoice].notes[n].ontime < (i+1) * e.harBlock \wedge \\
\qquad n \in \langle \cup m : 0 \le m < |\ e.ornamentalNotes\ | : e.ornamentalNotes[m].ornNoteNumber \rangle \wedge \\
\qquad [\![\ n, l\ ]\!] \in e.ornamentalNotes \wedge \\
\qquad 0 \le h < |FigTable(e,l)| \wedge FigTable(e,l)[h] = e.posFig[i].fig[k] : \\
\qquad\quad FigProbTable(e,l)[h] \\
\quad \rangle \\
\quad) \\
\rangle \\
\rangle
\end{array}
$$

The functions $SixFour$, $CadenceSubdominant$, $CadenceDominant$, and $CadenceTonic$ are specified by:

$$
\begin{array}{l}
SixFour(e) = \\
\quad \{x \mid 0 \le x < e.numMeasures * \frac{e.meter.num}{e.harBlock} \wedge \\
\qquad \langle \exists y, z : y \in \mathbb{Q}^+ \cup \{0\} \wedge z \in \mathbb{Q}^+ \wedge [\![\ y, z\ ]\!] \in e.sixFourChords \wedge y \le x * e.harBlock < z \rangle \\
\quad \}
\end{array}
$$

$$
\begin{array}{l}
CadenceSubdominant(e) = \\
\quad \{x \mid 0 \le x < e.numMeasures * \frac{e.meter.num}{e.harBlock} \wedge \\
\qquad \langle \exists y, z : y \in \mathbb{Q}^+ \cup \{0\} \wedge z \in \mathbb{Q}^+ \wedge [\![\ y, z\ ]\!] \in e.cadenceSubDom \wedge y \le x * e.harBlock < z \rangle \\
\quad \}
\end{array}
$$

$$
\begin{array}{l}
CadenceDominant(e) = \\
\quad \{x \mid 0 \le x < e.numMeasures * \frac{e.meter.num}{e.harBlock} \wedge \\
\qquad \langle \exists y, z : y \in \mathbb{Q}^+ \cup \{0\} \wedge z \in \mathbb{Q}^+ \wedge [\![\ y, z\ ]\!] \in e.cadenceDom \wedge y \le x * e.harBlock < z \rangle \\
\quad \}
\end{array}
$$

$$
\begin{array}{l}
CadenceTonic(e) = \\
\quad \{x \mid 0 \le x < e.numMeasures * \frac{e.meter.num}{e.harBlock} \wedge \\
\qquad \langle \exists y, z : y \in \mathbb{Q}^+ \cup \{0\} \wedge z \in \mathbb{Q}^+ \wedge [\![\ y, z\ ]\!] \in e.cadenceTonic \wedge y \le x * e.harBlock < z \rangle \\
\quad \}
\end{array}
$$

Rule C1P1R6 has the associated predicate $RuleC1P1R6$:

$RuleC1P1R6.i$:
$\langle \forall n : 1 \leq n < e.voices[e.numVoices - 1].numNotes \wedge$
$\quad i * e.harBlock \leq e.voices[e.numVoices - 1].notes[n].ontime < (i+1) * e.harBlock \wedge$
$\quad\quad e.voices[e.numVoices - 1].notes[n-1].ontime < i * e.harBlock :$
$\quad Q6.n$
$\rangle$

$Q6.n$:
$| e.voices[e.numVoices].notes[n].pitch.tcd -$
$\quad e.voices[e.numVoices - 1].notes[n-1].pitch.tcd \mid \in \{3, 4\}$
$\Rightarrow$
$\Big((e.figSymSeq[i] \neq e.figSymSeq[i-1]) \vee$
$\quad \Big(ChordNote($
$\quad\quad e.voices[e.numVoices - 1].notes[n].pitch.tcd,$
$\quad\quad e.voices[e.numVoices - 1].notes[n].pitch.acc,$
$\quad\quad e.figSymSeq[i]$
$\quad) = 3$
$\quad \wedge$
$\quad ChordNote($
$\quad\quad e.voices[e.numVoices - 1].notes[n-1].pitch.tcd,$
$\quad\quad e.voices[e.numVoices - 1].notes[n-1].pitch.acc,$
$\quad\quad e.figSymSeq[i-1]$
$\quad) = 3$
$\quad \Big)$
$\Big)$

This predicate uses function $ChordNote(tcd, acc : \mathbb{N}; f : FigSym) : \{0, 1, 3, 5, 7\}$. The return value equals 0 if tcd and acc do not represent a chord note of f, 1 if tcd and acc represent the root note of f, 3 in case of the third, 5 in case of the fifth and 7 in case of the seventh.

A.2.2 Writing the non-given outer voice

The function $NGOVNoteTable$ uses a function $Append(tcd, acc : \mathbb{Z}; e : Exercise; pitchArray : Pitch[\,])$ which appends the pitch (tcd, acc) to array of pitches $pitchArray$ if this pitch is within the voice range of the non-given outer voice of exercise e, and has this new pitch array as return value:

func $Append(tcd, acc : \mathbb{Z}; e : Exercise; pitchArray : Pitch[\,]) : Pitch[\,]$
$[\![$
$\quad$ **if** $e.givenVoice = 0 \rightarrow$
$\quad\quad$ **if** $60 \leq PitchToMidiNumber(tcd, acc, e.key) \leq 81 \rightarrow$
$\quad\quad\quad Append := pitchArray \triangleright [\![\, tcd, acc \,]\!]$
$\quad\quad [\!]$ $PitchToMidiNumber(tcd, acc, e.key) < 60 \vee$
$\quad\quad\quad\quad PitchToMidiNumber(tcd, acc, e.key) > 81 \rightarrow$
$\quad\quad\quad Append := pitchArray$
$\quad\quad$ **fi**
$\quad [\!]$ $\neg e.givenVoice = e.numVoices - 1 \rightarrow$
$\quad\quad$ **if** $40 \leq PitchToMidiNumber(tcd, acc, e.key) \leq 64 \rightarrow$

```
      Append : = pitchArray ▷ |[ tcd, acc ]|
    [] PitchToMidiNumber(tcd, acc, e.key) < 40 ∨
         PitchToMidiNumber(tcd, acc, e.key) > 64 →
      Append : = pitchArray
    fi
  fi
]|
```

The function $PitchToMidiNumber(tcd, acc : \mathbb{Z}; key : Key)$ computes and returns the midi note number of a note from its pitch (tcd, acc) and the key key in which it occurs. Middle c is represented by midi note number 60, and a note with midi note number $n + 1$ is one semitone higher than a note with number n. See [Ass01] for more details on MIDI conventions.

The function $NGOVNoteTable$ is given by:

```
func NGOVNoteTable(e : Exercise; i : ℕ) : Pitch[ ]
|[
  var j : ℤ; f : FigSym; accScale : ℤ[7];

  f, accScale, NGOVNoteTable : = e.figSymSeq[i], 0⃗, [ ];

  if Alteration(f) = pic → accScale[0] : = 1
  [] Alteration(f) = o → accScale[4] : = 1
  [] Alteration(f) = md → accScale[5] : = −1
  [] Alteration(f) = dm → accScale[5] : = 1
  [] Alteration(f) ∈ {∅, ♯5, ♮3} ∧ e.key.mode = 0 → accScale[3] : = 1; accScale[5] : = −1
  [] Alteration(f) ∈ {∅, ♯5, ♮3} ∧ e.key.mode = 5 → accScale[3] : = 1
  fi;

  { raising the leading tone in the minor mode: }
  if RomanNum(f) ∈ {5, 7} ∧ e.key.mode = 5 ∧ Alteration(f) ≠ aeol → accScale[6] : = 1
  [] RomanNum(f) ∉ {5, 7} ∨ e.key.mode ≠ 5 ∨ Alteration(f) = aeol → skip
  fi;

  if e.givenVoice = 0 →
    j : = −14 + RomanNum(f) − 1;

    do j ≤ 12 − e.key.whiteKey →
      { append root note of chord: }
      NGOVNoteTable : =
        Append(j, accScale[RomanNum(f) − 1 mod 7], e, NGOVNoteTable);
      { append third of chord: }
      j : = j + 2;
      NGOVNoteTable : =
        Append(j, accScale[RomanNum(f) + 1 mod 7], e, NGOVNoteTable);
      { append fifth of chord: }
      j : = j + 2;
      NGOVNoteTable : =
        Append(j, accScale[RomanNum(f) + 3 mod 7], e, NGOVNoteTable);
      { append seventh of chord: }
      j : = j + 2;
      if Inversion(f) ∈ {7, 6/5, 4/3, 2, ♯5, ♮3} →
```

$NGOVNoteTable :=$
$Append(j, accScale[RomanNum(f) + 5 \bmod 7], e, NGOVNoteTable)$
$[\!]\ Inversion(f) \notin \{7, {}^{6}_{5}, {}^{4}_{3}, 2, {}^{\emptyset}_{5}, {}^{4}_{3}\} \rightarrow$ skip
fi;
$j := j + 1$
od
$[\!]\ e.givenVoice = e.numVoices - 1 \rightarrow$
$j := -21 + RomanNum(f) - 1;$

{ choose another bass tone (instead of the root) in case of an inversion: }
if $Inversion(f) \in \{6, {}^{6}_{5}, {}^{\emptyset}_{5}\} \rightarrow j := j - 5$
$[\!]\ Inversion(f) \in \{{}^{6}_{4}, {}^{4}_{3}, {}^{4}_{3}\} \rightarrow j := j - 3$
$[\!]\ Inversion(f) = 2 \rightarrow j := j - 1$
$[\!]\ Inversion(f) \notin \{6, {}^{6}_{5}, {}^{\emptyset}_{5}, {}^{6}_{4}, {}^{4}_{3}, {}^{4}_{3}, 2\} \rightarrow$ skip
fi;

do $j \leq 2 - e.key.whiteKey \rightarrow$
$NGOVNoteTable :=$
$Append(j, accScale[RomanNum(f) - 1 \bmod 7], e, NGOVNoteTable);$
$j := j + 7$
od
fi
]|

The function $Alteration(f : FigSym) : \{pic, o, md, dm, aeol, \emptyset, unaltered, {}^{\emptyset}_{5}, {}^{4}_{3}\}$ returns the alteration of figuring symbol f, $RomanNum(f : FigSym) : \mathbb{N}$ returns only the numerical value of the roman numeral of figuring symbol f, and $Inversion(f : FigSym) : \{5, 6, {}^{6}_{5}, {}^{\emptyset}_{5}, {}^{6}_{4}, {}^{4}_{3}, {}^{4}_{3}, 7, 2\}$ returns the inversion of figuring symbol f.

Variable *accScale* stores, for each note of the scale, the corresponding Accidental Amount. For instance, in case of IV_{md}, the sixth of the scale is lowered, and hence $accScale[5]$ has to equal -1.

The role of the variable j is that of the *tcd*'s of the pitches that are to be appended to pitch array $NGOVNoteTable$. The bounds of the value of j ($-14 + RomanNum(f) - 1 \leq j \leq 12 - e.key.whiteKey$ for given bass and $-21 + RomanNum(f) - 1 \leq j \leq 2 - e.key.whiteKey$ for given soprano) ensure that the whole range of soprano and bass notes is used in constructing possible notes for the non-given outer voice.

$P10$ is given by:

$P10 :$
$e.givenVoice = 0 \Rightarrow$
$\langle \forall i : i \in HalfCadence(e) \Rightarrow$
$\langle \forall j : 0 \leq j < e.posNGOV[i].numPos :$
$\Big(\big(e.posNGOV[i].ngovNotes[j].tcd \in \{1 \bmod 7, 2 \bmod 7\} \wedge$
$e.posNGOV[i].ngovNotes[j].acc = 0 \big)$
$\equiv$
$e.posNGOV[i].prob[j] > 0$
$\Big)$
$\rangle \wedge$
$(e.posNGOV[i].prob = e.posNGOV[i].probBackup) \wedge$

$$
\begin{array}{l}
\quad \big\langle \sum j : 0 \le j < e.posNGOV[i].numPos : e.posNGOV[i].prob[j] \big\rangle = 1 \\
\big\rangle \wedge \\
\big\langle \forall i : 0 \le i < e.numMeasures * \frac{e.meter.num}{e.harBlock} \wedge PassingSixFourChord.i : \\
\quad \big\langle \exists j : 0 \le j < e.posNGOV[i].numPos \wedge \\
\qquad e.posNGOV[i].prob[j] > PROBPASSINGSIXFOUR \wedge \\
\qquad posNGOV[i].ngovNotes[j].pitch = [\![\ RomanNum(e.figSymSeq[i]) + 3 \bmod 7, 0\ ]\!] \\
\quad \big\rangle \wedge \\
\quad \big\langle \exists j : 0 \le j < e.posNGOV[PrevFigSym(e,i)].numPos \wedge \\
\qquad e.posNGOV[PrevFigSym(e,i)] > PROBPASSINGSIXFOUR \wedge \\
\qquad Inversion(PrevFigSym(e,i)) = 6 \Rightarrow \\
\qquad\quad posNGOV[PrevFigSym(e,i)].ngovNotes[j].pitch = \\
\qquad\qquad [\![\ RomanNum(e.figSymSeq[i]) - 1 \bmod 7, 0\ ]\!] \wedge \\
\qquad Inversion(PrevFigSym(e,i)) = 5 \Rightarrow \\
\qquad \Big(\\
\qquad\quad posNGOV[PrevFigSym(e,i)].ngovNotes[j].pitch = \\
\qquad\qquad [\![\ RomanNum(e.figSymSeq[i]) - 1 \bmod 7, 0\ ]\!] \\
\qquad\quad \vee \\
\qquad\quad \big(e.figSymSeq[PrevFigSym(e,i)] = \mathrm{IV}_{md} \wedge \\
\qquad\quad posNGOV[PrevFigSym(e,i)].ngovNotes[j].pitch = \\
\qquad\qquad [\![\ RomanNum(e.figSymSeq[i]) - 1 \bmod 7, -1\ ]\!]\big) \\
\qquad\quad \vee \\
\qquad\quad \big(e.figSymSeq[PrevFigSym(e,i)] = \mathrm{IV}_{dm} \wedge \\
\qquad\quad posNGOV[PrevFigSym(e,i)].ngovNotes[j].pitch = \\
\qquad\qquad [\![\ RomanNum(e.figSymSeq[i]) - 1 \bmod 7, 1\ ]\!]\big) \\
\qquad \Big) \\
\quad \big\rangle \wedge \\
\quad \big\langle \exists j : 0 \le j < e.posNGOV[NextFigSym(e,i)].numPos \wedge \\
\qquad e.posNGOV[NextFigSym(e,i)] > PROBPASSINGSIXFOUR \wedge \\
\qquad Inversion(NextFigSym(e,i)) = 6 \Rightarrow \\
\qquad\quad posNGOV[NextFigSym(e,i)].ngovNotes[j].pitch = \\
\qquad\qquad [\![\ RomanNum(e.figSymSeq[i]) - 1 \bmod 7, 0\ ]\!] \wedge \\
\qquad Inversion(NextFigSym(e,i)) = 5 \Rightarrow \\
\qquad \Big(\\
\qquad\quad posNGOV[NextFigSym(e,i)].ngovNotes[j].pitch = \\
\qquad\qquad [\![\ RomanNum(e.figSymSeq[i]) - 1 \bmod 7, 0\ ]\!] \\
\qquad\quad \vee \\
\qquad\quad \big(e.figSymSeq[NextFigSym(e,i)] = \mathrm{IV}_{md} \wedge \\
\qquad\quad posNGOV[NextFigSym(e,i)].ngovNotes[j].pitch = \\
\qquad\qquad [\![\ RomanNum(e.figSymSeq[i]) - 1 \bmod 7, -1\ ]\!]\big) \\
\qquad\quad \vee \\
\qquad\quad \big(e.figSymSeq[NextFigSym(e,i)] = \mathrm{IV}_{dm} \wedge \\
\qquad\quad posNGOV[NextFigSym(e,i)].ngovNotes[j].pitch = \\
\qquad\qquad [\![\ RomanNum(e.figSymSeq[i]) - 1 \bmod 7, 1\ ]\!]\big) \\
\qquad \Big) \\
\quad \big\rangle \\
\big\rangle
\end{array}
$$

The predicate $PassingSixFourChord.i$ says if the chord at harmony block i in exercise e is a passing six-four chord or not:

$PassingSixFourChord.i$:
$(e.figSymSeq[PrevFigSym(e,i)] = \mathrm{I} \wedge e.figSymSeq[i] = \mathrm{V}_4^6 \wedge$
$\quad e.figSymSeq[NextFigSym(e,i)] = \mathrm{I}^6) \vee$
$(e.figSymSeq[PrevFigSym(e,i)] = \mathrm{I}^6 \wedge e.figSymSeq[i] = \mathrm{V}_4^6 \wedge$
$\quad e.figSymSeq[NextFigSym(e,i)] = \mathrm{I}) \vee$
$(e.figSymSeq[PrevFigSym(e,i)] = \mathrm{IV} \wedge e.figSymSeq[i] = \mathrm{I}_4^6 \wedge$
$\quad e.figSymSeq[NextFigSym(e,i)] = \mathrm{IV}^6) \vee$
$(e.figSymSeq[PrevFigSym(e,i)] = \mathrm{IV}^6 \wedge e.figSymSeq[i] = \mathrm{I}_4^6 \wedge$
$\quad e.figSymSeq[NextFigSym(e,i)] = \mathrm{IV})$

The function $NextFigSym(e : Exercise; i : \mathbb{N})$ returns the number of the harmony block in which the figuring symbol that follows $e.figSymSeq[i]$ in exercise e can be found; $PrevFigSym(e : Exercise; i : \mathbb{N})$ returns the harmony block number of the previous one.

The value of $PROBPASSINGSIXFOUR$ corresponds to the probability that a bass line such as the one given in Figure 1.5 (page 17) to the left is coupled with a soprano voice given to the right. This value has to result from statistical analysis of existing, correctly filled in harmony exercises.

Lemma 1. *For any voice v, there is no note in v for which the sum of the ontime and the duration is nonnegative, iff the number of notes in v equals* 0.

Proof. We can write 'there is no note in v for which the sum of the ontime and the duration is nonnegative' as follows:

$\langle \nexists n : 0 \leq n < e.voices[v].numNotes :$
$\quad e.voices[v].notes[n].ontime + e.voices[v].notes[n].duration \geq 0 \rangle$

The following derivation now proves the lemma:

$\langle \nexists n : 0 \leq n < e.voices[v].numNotes :$
$\quad e.voices[v].notes[n].ontime + e.voices[v].notes[n].duration \geq 0 \rangle$

$\equiv$ { De Morgan }

$\langle \forall n : 0 \leq n < e.voices[v].numNotes :$
$\quad e.voices[v].notes[n].ontime + e.voices[v].notes[n].duration < 0 \rangle$

$\equiv$ { $ontime, duration \in \mathbb{Q}^+ \cup \{0\} \Rightarrow ontime + duration \geq 0$ }

$\langle \forall n : 0 \leq n < e.voices[v].numNotes : false \rangle$

$\equiv$ { domain must be empty }

$e.voices[v].numNotes \leq 0$

$\equiv$ { $numNotes \in \mathbb{N}$ }

$e.voices[v].numNotes = 0$

□

Rules C1P0R8 and C1P1R11 are expressed by the predicates $RuleC1P0R8$ and $RuleC1P1R11$ respectively:

$$
\begin{array}{l}
RuleC1P0R8.i : \\
\langle \forall n : 0 \leq n < e.voices[0].numNotes : \\
\quad 40 \leq PitchToMidiNumber(e.voices[0].notes[n].pitch.tcd, \\
\qquad e.voices[0].notes[n].pitch.acc, e.key) \leq 64 \\
\rangle
\end{array}
$$

$$
\begin{array}{l}
RuleC1P1R11.i : \\
\langle \forall n : 0 < n < e.voices[0].numNotes : \\
\quad |PitchToMidiNumber(e.voices[0].notes[n].pitch.tcd, \\
\qquad e.voices[0].notes[n].pitch.acc, e.key) - \\
\quad PitchToMidiNumber(e.voices[0].notes[n-1].pitch.tcd, \\
\qquad e.voices[0].notes[n-1].pitch.acc, e.key)| \leq 12 \\
\rangle
\end{array}
$$

Rule C1P1R2SB is expressed by the following predicate:

$$
\begin{array}{l}
RuleC1P1R2SB.i : \\
CheckParallelMotion.i.(e.numVoices - 1 - e.givenVoice).(e.givenVoice).0 \wedge \\
CheckParallelMotion.i.(e.numVoices - 1 - e.givenVoice).(e.givenVoice).7
\end{array}
$$

$$
\begin{array}{l}
CheckParallelMotion.i.v0.v1.interval : \\
\langle \forall n : 0 < n < e.voices[v0].numNotes : \\
\qquad i * e.harBlock \leq e.voices[v0].notes[n].ontime < (i+1) * e.harBlock \vee \\
\qquad (e.voices[v0].notes[n].ontime < i * harBlock \wedge \\
\qquad\quad e.voices[v0].notes[n].ontime + e.voices[v0].notes[n].duration > i * harBlock) \\
\quad : \\
\quad PitchToMidiNumber(\\
\qquad e.voices[v0].notes[n].pitch.tcd, \\
\qquad e.voices[v0].notes[n].pitch.acc, \\
\qquad e.key) - \\
\quad PitchToMidiNumber(\\
\qquad GetNoteAtTime(e, e.voices[v0].notes[n].ontime \uparrow i * e.harBlock, v1).pitch.tcd, \\
\qquad GetNoteAtTime(e, e.voices[v0].notes[n].ontime \uparrow i * e.harBlock, v1).pitch.acc, \\
\qquad e.key) \\
\quad = interval \bmod 12 \Rightarrow \\
\quad \Big(\\
\qquad PitchToMidiNumber(\\
\qquad\quad GetNoteAtTime(e, T, v0).pitch.tcd, \\
\qquad\quad GetNoteAtTime(e, T, v0).pitch.acc, \\
\qquad\quad e.key) - \\
\qquad PitchToMidiNumber(\\
\qquad\quad GetNoteAtTime(e, T, v1).pitch.tcd, \\
\qquad\quad GetNoteAtTime(e, T, v1).pitch.acc, \\
\qquad\quad e.key) \\
\qquad \neq interval \bmod 12 \vee
\end{array}
$$

$$
\begin{array}{l}
\quad e.voices[v0].notes[n].pitch = e.voices[v0].notes[n-1].pitch \\
\quad \big) \\
\big\rangle
\end{array}
$$

$$
\begin{array}{l}
T : e.voices[v0].notes[n-1] \uparrow \\
\quad \big\langle \uparrow m : 0 \leq m < e.voices[v1].numNotes \wedge \\
\qquad e.voices[v1].notes[m].ontime < e.voices[v0].notes[n].ontime \uparrow i * e.harBlock : \\
\qquad\quad e.voices[v1].notes[m].ontime \big\rangle
\end{array}
$$

Here, the function $GetNoteAtTime(e : Exercise; t : \mathbb{Q}^+ \cup \{0\}; voice : \mathbb{N})$ returns the note in voice *voice* at time t in exercise e. For the timing convention used, see the explanation of the data type *Note* on page 28.

$CheckParallelMotion.i.v0.v1.interval.log$ says that for all notes n that sound in voice $v0$ during harmony block i (except for the first note in $v0$), there must be no parallel motion of interval *interval* between voices $v0$ and $v1$. This is checked as follows. If the difference between the midi note numbers of note n in voice $v0$ and the note sounding simultaneously in voice $v1$ equals *interval* mod 12, then the difference between the note numbers of the notes that come before this moment (at a certain moment T) must not also equal *interval* mod 12, unless the pitches have not changed. The moment T is given by the maximum of the ontimes of the previous notes in both voices.

We need to use midi note numbers here, because using Tone Center Displacement would result in error messages in case, for instance, a parallel motion of non-perfect fifths has been written, which is allowed. For the same reason, the argument *interval* also needs to be used in terms of midi note numbers.

A.2.3 Writing the inner voices

Rule C1P2R3 is expressed by predicate $RuleC1P2R3$:

$$
\begin{array}{l}
RuleC1P2R3.e : \\
\big\langle \forall i : 0 \leq i < e.numMeasures * \frac{e.meter.num}{e.harBlock} : \\
\quad \big(e.figSymSeq[i] = \mathrm{IV}^6 \wedge NextFigSym(e,i) = \mathrm{V}^6 \wedge e.key.mode = 0 \\
\qquad \Rightarrow \\
\qquad\quad e.posDup[i].numPos = 2 \wedge e.posDup[i][0] = \\
\qquad\qquad [1,3,5, \\
\qquad\qquad\quad ChordNote(\\
\qquad\qquad\qquad GetNoteAtHarBlock(i, e.numVoices - 1).pitch.tcd, \\
\qquad\qquad\qquad GetNoteAtHarBlock(i, e.numVoices - 1).pitch.acc, \\
\qquad\qquad\qquad e.figSymSeq[i] \\
\qquad\qquad\quad) \\
\qquad\qquad] \wedge \\
\qquad\quad e.posDup[i][1] = [1,3,3,5] \\
\quad \big) \wedge \\
\quad \big(e.figSymSeq[i] = \mathrm{IV}^6_{dm} \wedge NextFigSym(e,i) = \mathrm{V}^6 \wedge e.key.mode = 5 \\
\qquad \Rightarrow \\
\qquad\quad e.posDup[i].numPos = 2 \wedge e.posDup[i][0] = \\
\qquad\qquad [1,3,5, \\
\qquad\qquad\quad ChordNote(\\
\qquad\qquad\qquad GetNoteAtHarBlock(i, e.numVoices - 1).pitch.tcd, \\
\qquad\qquad\qquad GetNoteAtHarBlock(i, e.numVoices - 1).pitch.acc,
\end{array}
$$

$$
\begin{array}{l}
\qquad\qquad\qquad\quad e.figSymSeq[i] \\
\qquad\qquad\qquad\quad) \\
\qquad\qquad\quad] \wedge \\
\qquad\qquad e.posDup[i][1] = [1,3,5,5] \\
\qquad)
\end{array}
$$
$$\rangle$$

Rule C1P3R6 is expressed by predicate $RuleC1P3R6$:

$$
\begin{array}{l}
RuleC1P3R6.e : \\
\langle \forall i : 0 \leq i < e.numMeasures * \frac{e.meter.num}{e.harBlock} : \\
\qquad \Big(\\
\qquad\quad (\\
\qquad\qquad (e.figSymSeq[i] = \mathrm{V}_4^6 \wedge e.figSymSeq[PrevFigSym(e,i)] = \mathrm{I} \wedge \\
\qquad\qquad\quad e.figSymSeq[NextFigSym(e,i)] = \mathrm{I}^6) \vee \\
\qquad\qquad (e.figSymSeq[i] = \mathrm{V}_4^6 \wedge e.figSymSeq[PrevFigSym(e,i)] = \mathrm{I}^6 \wedge \\
\qquad\qquad\quad e.figSymSeq[NextFigSym(e,i)] = \mathrm{I}) \vee \\
\qquad\qquad (e.figSymSeq[i] = \mathrm{I}_4^6 \wedge e.figSymSeq[PrevFigSym(e,i)] = \mathrm{IV} \wedge \\
\qquad\qquad\quad e.figSymSeq[NextFigSym(e,i)] = \mathrm{IV}^6) \vee \\
\qquad\qquad (e.figSymSeq[i] = \mathrm{I}_4^6 \wedge e.figSymSeq[PrevFigSym(e,i)] = \mathrm{IV}^6 \wedge \\
\qquad\qquad\quad e.figSymSeq[NextFigSym(e,i)] = \mathrm{IV}) \\
\qquad\quad) \wedge \\
\qquad\quad (\\
\qquad\qquad (GetNoteAtHarBlock(e,i,e.numVoices-1).tcd - \\
\qquad\qquad\quad GetNoteAtHarBlock(e,PrevFigSym(e,i),e.numVoices-1).tcd = 2 \wedge \\
\qquad\qquad GetNoteAtHarBlock(e,NextFigSym(e,i),e.numVoices-1).tcd - \\
\qquad\qquad\quad GetNoteAtHarBlock(e,i,e.numVoices-1).tcd = 3) \vee \\
\qquad\qquad (GetNoteAtHarBlock(e,i,e.numVoices-1).tcd - \\
\qquad\qquad\quad GetNoteAtHarBlock(e,PrevFigSym(e,i),e.numVoices-1).tcd = -3 \wedge \\
\qquad\qquad GetNoteAtHarBlock(e,NextFigSym(e,i),e.numVoices-1).tcd - \\
\qquad\qquad\quad GetNoteAtHarBlock(e,i,e.numVoices-1).tcd = -2) \\
\qquad\quad) \\
\qquad \Big) \Rightarrow \\
\qquad e.posDup[i].numPos = 1 \wedge e.posDup[i].dup[0] = [1,1,3,5]
\end{array}
$$
$$\rangle$$

Rule C1P3R4 is expressed by predicate $RuleC1P3R4$. This predicate uses a predicate $CadentialSixFourChord.e.j$, saying if the chord at harmony block j in exercise e is a cadential six-four chord or not:

$$
\begin{array}{l}
CadentialSixFourChord.e.j : \\
\big(e.figSymSeq[j] = \mathrm{I}_4^6 \wedge e.figSymSeq[NextFigSym(e,j)] = \mathrm{V} \wedge \\
\qquad (e.figSymSeq[PrevFigSym(e,j)] = \mathrm{V} \Rightarrow \\
\qquad\quad Stress(e, j * e.harBlock) < Stress(e, NextFigSym(e,j) * e.harBlock) \\
\qquad) \\
\big) \\
\vee \\
\big(e.figSymSeq[j] = \mathrm{IV}_4^6 \wedge e.figSymSeq[NextFigSym(e,j)] = \mathrm{I} \wedge
\end{array}
$$

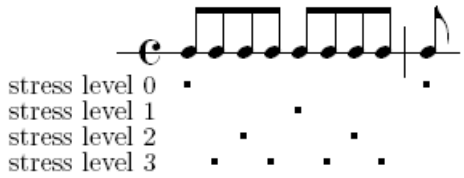

Figure A.1: Metrical accent in $\frac{4}{4}$ time

$$
\begin{array}{l}
\quad (e.figSymSeq[PrevFigSym(e,j)] = \mathrm{I} \Rightarrow \\
\qquad Stress(e, j * e.harBlock) < Stress(e, NextFigSym(e,j) * e.harBlock) \\
\quad) \\
)
\end{array}
$$

The function $Stress(e : Exercise; t : \mathbb{Q}^+ \cup \{0\}) : \mathbb{N}$ captures the notion of strong and weak beats. The return value is a natural denoting the amount of stress at time t in exercise e. Regarding stress amount, we propose the following convention, although this is admittedly open for discussion. The first beat of every measure has stress 0, and the less stress a beat has, the higher the stress number, as demonstrated in Figure A.1. This convention is derived from the notion of metrical accent, used by F. Lerdahl and R.S. Jackendoff in [LJ83].

Below we give a specification of the function *Stress* in case the numerator of the meter is a power of 2. We restrict ourselves to this meter type for the moment. The postcondition of $Stress(e : Exercise; t : \mathbb{Q}^+ \cup \{0\})$ is then given by:

$$
\begin{array}{l}
\langle \exists n : n \geq 0 \wedge j = n * e.key.num \rangle \equiv Stress = 0 \\
\wedge \\
\langle \exists q : q \in \mathbb{N}^+ \wedge e.meter.num = 2^q \rangle \Rightarrow \\
\quad \langle \exists n, k : n, k \geq 1 \wedge j = k * (\frac{1}{2})^{n - \log_2(e.meter.num)} \rangle \equiv \\
\qquad Stress = \langle \downarrow n : n \geq 1 \wedge \langle \exists k : k \geq 1 \wedge j = k * (\frac{1}{2})^{n - \log_2(e.meter.num)} \rangle : n \rangle
\end{array}
$$

$RuleC1P3R4$ is can now be given by:

$$RuleC1P3R4 : \langle \forall j : 0 \leq j \leq i \wedge CadentialSixFourChord.e.j : Q7 \vee (Q8 \wedge Q9) \rangle$$

$Q7$ expresses that if some note n has the root or third of a cadential six-four chord, the note following n needs to be a second lower than n:

$$
\begin{array}{l}
Q7 : \\
\langle \forall v, n : 0 \leq v < e.numVoices \wedge 0 \leq n < e.voices[v].numNotes \wedge \\
\qquad e.voices[v].notes[n].ontime < e.harBlock * NextFigSym(e,j) \wedge \\
\qquad \neg \langle \exists n' : n < n' < e.voices[v].numNotes \wedge \\
\qquad\qquad voices[v].notes[n'].ontime < e.harBlock * NextFigSym(e,j) \rangle \wedge \\
\qquad ChordNote(\\
\qquad\quad e.voices[v].notes[n].tcd,
\end{array}
$$

$$
\begin{array}{l}
\quad\quad e.voices[v].notes[n].acc, \\
\quad\quad e.figSymSeq[j] \\
\quad) \in \{1,3\} : \\
e.voices[v].notes[n].tcd + 1 = e.voices[v].notes[n+1].tcd \\
\rangle
\end{array}
$$

$Q8$ and $Q9$ say that the root and the third of the chord respectively lie in one of the inner voices:

$$
\begin{array}{l}
Q8 : \\
\langle \exists v : 0 < v < e.numVoices - 1 \wedge \\
\quad \langle \forall n : 0 \leq n < e.voices[v].numNotes \wedge \\
\quad\quad e.voices[v].notes[n].ontime < e.harBlock * NextFigSym(e,j) \wedge \\
\quad\quad \neg \langle \exists n' : n < n' < e.voices[v].numNotes \wedge \\
\quad\quad\quad voices[v].notes[n'].ontime < e.harBlock * NextFigSym(e,j) \rangle : \\
\quad ChordNote(\\
\quad\quad e.voices[v].notes[n].tcd, \\
\quad\quad e.voices[v].notes[n].acc, \\
\quad\quad e.figSymSeq[j] \\
\quad) = 1 \\
\quad \rangle \\
\rangle
\end{array}
$$

$$
\begin{array}{l}
Q9 : \\
\langle \exists v : 0 < v < e.numVoices - 1 \wedge \\
\quad \langle \forall n : 0 \leq n < e.voices[v].numNotes \wedge \\
\quad\quad e.voices[v].notes[n].ontime < e.harBlock * NextFigSym(e,j) \wedge \\
\quad\quad \neg \langle \exists n' : n < n' < e.voices[v].numNotes \wedge \\
\quad\quad\quad voices[v].notes[n'].ontime < e.harBlock * NextFigSym(e,j) \rangle : \\
\quad ChordNote(\\
\quad\quad e.voices[v].notes[n].tcd, \\
\quad\quad e.voices[v].notes[n].acc, \\
\quad\quad e.figSymSeq[j] \\
\quad) = 3 \\
\quad \rangle \\
\rangle
\end{array}
$$

$Q11$ evaluates to *true* iff a duplication of chord tones is chosen that does not occur in $e.posDup[i].dup$. To this end, it is checked if the multiset of chord notes of this note, and the other three notes in the other voices sounding simultaneously, is an element of $e.posDup[i].dup$ (i.e. an allowed duplication) or not. In order to find the notes that sound simultaneously with note n of voice v, we use the function $GetNoteAtTime$, using $e.voices[v].notes[n].ontime \uparrow i * e.harBlock$ as second argument; this is needed in case the ontime of note n comes before harmony block i. $Q11$ assumes exercise e to contain four voices (standard four-part harmony):

$$
\begin{array}{l}
Q11.i.v.n : \\
[\\
\quad ChordNote(\\
\quad\quad e.voices[v].notes[n].pitch.tcd, \\
\quad\quad e.voices[v].notes[n].pitch.acc,
\end{array}
$$

$$
\begin{array}{l}
\quad e.figSymSeq[i] \\
\quad), \\
\quad ChordNote(\\
\quad\quad GetNoteAtTime(e, e.voices[v].notes[n].ontime \uparrow i * e.harBlock, \\
\quad\quad\quad v + 1 \bmod e.numVoices).pitch.tcd, \\
\quad\quad GetNoteAtTime(e, e.voices[v].notes[n].ontime \uparrow i * e.harBlock, \\
\quad\quad\quad v + 1 \bmod e.numVoices).pitch.acc, \\
\quad\quad e.figSymSeq[i] \\
\quad), \\
\quad ChordNote(\\
\quad\quad GetNoteAtTime(e, e.voices[v].notes[n].ontime \uparrow i * e.harBlock, \\
\quad\quad\quad v + 2 \bmod e.numVoices).pitch.tcd, \\
\quad\quad GetNoteAtTime(e, e.voices[v].notes[n].ontime \uparrow i * e.harBlock, \\
\quad\quad\quad v + 2 \bmod e.numVoices).pitch.acc, \\
\quad\quad e.figSymSeq[i] \\
\quad), \\
\quad ChordNote(\\
\quad\quad GetNoteAtTime(e, e.voices[v].notes[n].ontime \uparrow i * e.harBlock, \\
\quad\quad\quad v + 3 \bmod e.numVoices).pitch.tcd, \\
\quad\quad GetNoteAtTime(e, e.voices[v].notes[n].ontime \uparrow i * e.harBlock, \\
\quad\quad\quad v + 3 \bmod e.numVoices).pitch.acc, \\
\quad\quad e.figSymSeq[i] \\
\quad) \\
] \notin e.posDup[i].dup \\
\wedge \\
\langle \forall w : 1 \leq w < e.numVoices - 1 : \\
\quad ChordNote(\\
\quad\quad e.voices[w].notes[n].pitch.tcd, \\
\quad\quad e.voices[w].notes[n].pitch.acc, \\
\quad\quad e.figSymSeq[i] \\
\quad) \neq 0 \\
\rangle
\end{array}
$$

A.3 Checking harmony

This section contains the details concerning Chapter 6, Checking harmony.

A.3.1 Checking figuring

Rule C1P1R6 has the following predicate for the message log:

$$
\begin{array}{l}
LogRuleC1P1R6.i.log : \\
\langle \forall n : 1 \leq n < e.voices[e.numVoices - 1].numNotes \wedge \\
\quad i * e.harBlock \leq e.voices[e.numVoices - 1].notes[n].ontime < (i + 1) * e.harBlock \wedge \\
\quad e.voices[e.numVoices - 1].notes[n - 1] < i * e.harBlock \wedge \neg Q6 : \\
\quad\quad \langle \exists j : 0 \leq j \wedge log[j] = [\![\ error; i * e.harBlock; \\
\quad\quad\quad\quad\quad\quad \text{“leap in soprano not harmonised with one chord”} \ ]\!] \rangle \\
\rangle
\end{array}
$$

A.3.2 Checking outer voices

Rule C1P0R8 has the following predicate for the message log:

$$
\begin{array}{l}
LogRuleC1P0R8.i.log: \\
\langle \forall n : 1 \leq n < e.voices[0].numNotes \wedge \\
\quad i * e.harBlock \leq e.voices[0].notes[n].ontime < (i+1) * e.harBlock \wedge \\
\quad (\\
\qquad PitchToMidiNumber(\\
\qquad\quad e.voices[0].notes[n].pitch.tcd, \\
\qquad\quad e.voices[0].notes[n].pitch.acc, \\
\qquad\quad e.key \\
\qquad) > 64 \vee \\
\qquad PitchToMidiNumber(\\
\qquad\quad e.voices[0].notes[n].pitch.tcd, \\
\qquad\quad e.voices[0].notes[n].pitch.acc, \\
\qquad\quad e.key \\
\qquad) < 40 \\
\quad) : \\
\quad \langle \exists j : 0 \leq j \wedge log[j] = |[\ error; i * e.harBlock; \text{“bass voice out of range”}\ ]| \rangle \\
\rangle
\end{array}
$$

Predicate $LogParallelMotion.i.v0.v1.interval.log$ checks for parallels of the interval *interval* between voices $v0$ and $v1$ up to and including harmony block i in exercise e. Using this predicate, we can define the predicate $LogRuleC1P1R2SB$ for the message log concering rule C1P1R2SB:

$$
\begin{array}{l}
LogRuleC1P1R2SB.i.log: \\
LogParallelMotion.i.(e.givenVoice).(e.numVoices - 1 - e.givenVoice).0.log \wedge \\
LogParallelMotion.i.(e.givenVoice).(e.numVoices - 1 - e.givenVoice).7.log
\end{array}
$$

$$
\begin{array}{l}
LogParallelMotion.i.v0.v1.interval.log: \\
\langle \forall n : 0 < n < e.voices[v0].numNotes \wedge \\
\qquad i * e.harBlock \leq e.voices[v0].notes[n].ontime < (i+1) * e.harBlock \vee \\
\qquad (e.voices[v0].notes[n].ontime < i * harBlock \wedge \\
\qquad\quad e.voices[v0].notes[n].ontime + e.voices[v0].notes[n].duration > i * harBlock) \\
\quad : \\
\quad \Big(\\
\qquad PitchToMidiNumber(\\
\qquad\quad e.voices[v0].notes[n].pitch.tcd, \\
\qquad\quad e.voices[v0].notes[n].pitch.acc, \\
\qquad\quad e.key) - \\
\qquad PitchToMidiNumber(\\
\qquad\quad GetNoteAtTime(e, e.voices[v0].notes[n].ontime \uparrow i * e.harBlock, v1).pitch.tcd, \\
\qquad\quad GetNoteAtTime(e, e.voices[v0].notes[n].ontime \uparrow i * e.harBlock, v1).pitch.acc, \\
\qquad\quad e.key) \\
\qquad = interval \bmod 12 \wedge \\
\qquad PitchToMidiNumber(\\
\qquad\quad GetNoteAtTime(e, T, v0).pitch.tcd, \\
\qquad\quad GetNoteAtTime(e, T, v0).pitch.acc, \\
\qquad\quad e.key) -
\end{array}
$$

$$
\begin{array}{l}
\quad\quad PitchToMidiNumber(\\
\quad\quad\quad GetNoteAtTime(e, T, v1).pitch.tcd, \\
\quad\quad\quad GetNoteAtTime(e, T, v1).pitch.acc, \\
\quad\quad\quad e.key) \\
\quad\quad = interval \bmod 12 \wedge \\
\quad\quad (e.voices[v0].notes[n].pitch \neq e.voices[v0].notes[n-1].pitch) \\
\quad\big) \\
\quad\Rightarrow \\
\quad\langle \exists k : 0 \leq k \wedge log[k] = [\![\ error; i * e.harBlock;\ \text{“illegal parallel motion} \\
\quad\quad\quad\quad\quad\quad\quad\quad \text{between voices ”} \triangleright v0 \triangleright \text{“ and ”} \triangleright v1\ ]\!]\rangle \\
\rangle
\end{array}
$$

$$
\begin{array}{l}
T : e.voices[v0].notes[n-1] \uparrow \\
\quad\langle \uparrow m : 0 \leq m < e.voices[v1].numNotes \wedge \\
\quad\quad e.voices[v1].notes[m].ontime < e.voices[v0].notes[n].ontime \uparrow i * e.harBlock : \\
\quad\quad\quad e.voices[v1].notes[m].ontime\rangle
\end{array}
$$

$LogParallelMotion.i.v0.v1.interval.log$ says that for all notes n that sound in voice $v0$ during harmony block i (except for the first note in $v0$), if there is parallel motion between $v0$ and $v1$, an error message is added to log. This parallel motion is detected as follows. If the difference between the midi note numbers of note n in voice $v0$ and the note sounding simultaneously in voice $v1$ equals $interval$ mod 12, and the difference between the note numbers of the notes that come before this moment (at a certain moment T) also equals $interval$ mod 12, while the pitches have changed, we have detected a parallel motion of interval $interval$. The moment T is given by the maximum of the ontimes of the previous notes in both voices.

Again, we need to use midi note numbers, for reasons explained on page 86.

A.3.3 Checking inner voices

Rule C1P3R4 has the following predicate for the message log:

$$
\begin{array}{l}
LogRuleC1P3R4.i.log : \\
\langle \forall j : 0 \leq j \leq i \wedge CadentialSixFourChord.e.j : \\
\quad \neg Q7 \wedge (\neg Q8 \vee \neg Q9) \\
\quad\Rightarrow \\
\quad\langle \exists k : 0 \leq k \wedge log[k] = [\![\ error; j * e.harBlock; \text{“the root and third of a} \\
\quad\quad\quad\quad\quad\quad\quad\quad \text{cadential six-four chord must descend stepwise”}\ ]\!]\rangle \\
\rangle
\end{array}
$$

Appendix B

Listed harmony rules

This appendix lists the harmony rules as given in the introduction and the first three sections of [dCM]. Every rule is given a unique identifier and is categorized as either a strict rule, a soft rule, or a heuristic.

B.1 Introduction

This section lists the harmony rules of the introduction of [dCM]. We have given each rule a unique identifier.

B.1.1 Strict rules

Rule number	C1P0R0
Description	The distance between soprano and alto must not be greater than an octave.
Covered by	$AllRulesInnerVoices$, $LogVoiceLeadingRulesInnerVoices$

Rule number	C1P0R1
Description	The distance between alto and tenor must not be greater than an octave.
Covered by	$AllRulesInnerVoices$, $LogVoiceLeadingRulesInnerVoices$

Rule number	C1P0R2
Description	Voice-crossing is not allowed.
Covered by	$AllRulesNGOV$, $AllRulesInnerVoices$ $LogVoiceLeadingRulesOuterVoices$, $LogVoiceLeadingRulesInnerVoices$

Rule number	C1P0R3
Description	A dominant chord must not be followed by a subdominant chord.
Covered by	$AllRulesFiguring$, $LogFiguringRules$

Rule number	C1P0R4
Description	The distance between soprano and bass must be at least a fifth.
Covered by	$AllRulesNGOV$, $LogVoiceLeadingRulesOuterVoices$

Rule number C1P0R9
Description In case of a given soprano, the first note of the scale followed by the second must not be harmonised with IV–V.
Covered by *AllRulesFiguring*, *LogFiguringRules*

Rule number C1P0R10
Description In case of a given soprano, the fourth note of the scale followed by the fifth must not be harmonised with IV–V.
Covered by *AllRulesFiguring*, *LogFiguringRules*

B.1.2 Soft rules

Rule number C1P0R5
Description Soprano range: C4–A5
Covered by *AllRulesNGOV*, *LogVoiceLeadingRulesOuterVoices*

Rule number C1P0R6
Description Alto range: G3–E5
Covered by *AllRulesInnerVoices*, *LogVoiceLeadingRulesInnerVoices*

Rule number C1P0R7
Description Tenor range: C3–A4
Covered by *AllRulesInnerVoices*, *LogVoiceLeadingRulesInnerVoices*

Rule number C1P0R8
Description Bass range: E2–E4
Covered by *AllRulesNGOV*, *LogVoiceLeadingRulesOuterVoices*

B.2 Section 1 Main triads in root position

This section lists the harmony rules of the first section of [dCM]:

B.2.1 Strict rules

Rule number C1P1R0
Description In main triads in root position, the root note has to be duplicated.
Covered by *P*11

Rule number C1P1R1
Description The chord progression V-IV is not allowed.
Covered by *AllRulesFiguring*, *LogFiguringRules*

Rule number C1P1R2
Description Parallel octaves and parallel fifths are not allowed.
Covered by $AllRulesNGOV$, $AllRulesInnerVoices$
$LogVoiceLeadingRulesOuterVoices$, $LogVoiceLeadingRulesInnerVoices$

Rule number C1P1R3
Description When IV and IV_{MD} follow each other directly, the non-altered and the altered tone have to follow each other in the same voice.
Covered by $AllRulesNGOV$, $AllRulesInnerVoices$
$LogVoiceLeadingRulesOuterVoices$, $LogVoiceLeadingRulesInnerVoices$

Rule number C1P1R4
Description In the progression IV_{MD}–V, the lowered sixth has to resolve in the root of V.
Covered by $AllRulesNGOV$, $AllRulesInnerVoices$
$LogVoiceLeadingRulesOuterVoices$, $LogVoiceLeadingRulesInnerVoices$

Rule number C1P1R5
Description In the progression V_o–I, the raised fifth has to resolve in the third of I.
Covered by $AllRulesNGOV$, $AllRulesInnerVoices$
$LogVoiceLeadingRulesOuterVoices$, $LogVoiceLeadingRulesInnerVoices$

Rule number C1P1R6
Description A leap in the soprano greater than a third has to be harmonized with one and the same chord, unless it is a fourth/fifth leap from the third of one chord to the third of another chord.
Covered by $AllRulesFiguring$, $LogFiguringRules$

Rule number C1P1R7
Description Exercises end with IV–V–I.
Covered by $AllRulesFiguring$, $LogFiguringRules$

Rule number C1P1R8
Description Change of chords is compulsory from a weak to a strong beat, except after a caesura.
Covered by $AllRulesFiguring$, $LogFiguringRules$

Rule number C1P1R9
Description In the bass line, two great leaps in the same direction are not allowed.
Covered by $AllRulesNGOV$, $LogVoiceLeadingRulesOuterVoices$

Rule number C1P1R10
Description In the bass line, octave leaps have to be preceded and succeeded by contrary movement.
Covered by $AllRulesNGOV$, $LogVoiceLeadingRulesOuterVoices$

Rule number C1P1R11
Description In the bass line, a leap greater than an octave is not allowed.
Covered by *AllRulesNGOV*, *LogVoiceLeadingRulesOuterVoices*

Rule number C1P1R12
Description At a half cadence, the second or seventh note should preferably be in the soprano.
Covered by *AnalyzeFiguringAndGivenVoice*, *LogVoiceLeadingRulesOuterVoices*

B.2.2 Soft rules

Rule number C1P1R13
Description The range of the soprano should be limited.
Covered by *AllRulesNGOV*, *LogVoiceLeadingRulesOuterVoices*

Rule number C1P1R14
Description After a leap in the soprano, it is preferable to continue in the opposite direction.
Covered by *AllRulesNGOV*, *LogVoiceLeadingRulesOuterVoices*

Rule number C1P1R15
Description The soprano ends preferably on the root note.
Covered by *AllRulesNGOV*, *LogVoiceLeadingRulesOuterVoices*

Rule number C1P1R16
Description Tone repetitions from a weak to a strong beat should be avoided.
Covered by *AllRulesNGOV*, *LogVoiceLeadingRulesOuterVoices*

Rule number C1P1R17
Description The leading note in the soprano must resolve stepwise upward.
Covered by *AllRulesNGOV*, *LogVoiceLeadingRulesOuterVoices*

B.2.3 Heuristics

Rule number C1P1R18
Description In chord progressions with two main triads in root position, one should not go from a close spacing to an open spacing except in the following cases:
a the harmony does not change,
b the two chords are fifth-related and one of the middle voices (preferably the alto) has a common tone,
c the progression is IV–V in which the melody descends a seventh.

B.3 Section 2 Main triads in first inversion

This section lists the harmony rules of the second section of [dCM]:

B.3.1 Strict rules

Rule number C1P2R0
Description In main triads in first inversion, the melody note has to be duplicated.
Covered by $P11$

Rule number C1P2R1
Description In V^6, the third must not be duplicated.
Covered by $P11$

Rule number C1P2R2
Description Covered parallels are not allowed.
Covered by $AllRulesNGOV$, $LogVoiceLeadingRulesOuterVoices$

Rule number C1P2R3
Description In the progression IV^6–V^6 in the major mode and IV^6_{DM}–V^6 in the minor mode, the soprano, alto and bass form parallel sixth chords. Therefore the third may be duplicated in IV^6 and the fifth in IV^6_{DM}.
Covered by $P12$

Rule number C1P2R4
Description The progression V^6–IV is not allowed. The progression V^6–IV^6 is, but only if V^6 follows.
Covered by $AllRulesFiguring$, $LogFiguringRules$

Rule number C1P2R5
Description The progression IV^6–V^6 is not allowed in the minor mode. The progression IV^6_{DM}–V^6 is.
Covered by $AllRulesFiguring$, $LogFiguringRules$

Rule number C1P2R6
Description The progression V^6–IV^6 is not allowed in the minor mode. The progression V^6_{aeol}–IV^6 is.
Covered by $AllRulesFiguring$, $LogFiguringRules$

Rule number C1P2R7
Description In major keys, the raised fourth note always occurs in combination with the lowered sixth note.
Covered by $AllRulesFiguring$, $LogFiguringRules$

Rule number C1P2R8
Description In the Italian sixth chord, the fifth has to be duplicated.
Covered by $P11$

Rule number C1P2R9
Description Sixth chords are not allowed:
a in a half cadence,
b in V–I in the final cadence,
c if the third of the chord is in the given soprano.
Covered by *AllRulesFiguring*, *LogFiguringRules*

Rule number C1P2R10
Description A leap of a fourth/fifth or sixth from a weak to a strong beat in the soprano has to be harmonized with two chords, of which at least one should be a sixth chord.
Covered by *AllRulesFiguring*, *LogFiguringRules*

Rule number C1P2R11
Description The leading note in the bass should resolve in the root note, either directly (V^6–I) or indirectly (V^6–V–I).
Covered by *AllRulesFiguring*, *LogFiguringRules*

B.3.2 Soft rules

Rule number C1P2R12
Description With main triads in first inversion, it is mildly preferably to have the root note in the soprano.
Covered by *AllRulesNGOV*, *LogVoiceLeadingRulesOuterVoices*

B.3.3 Heuristics

Rule number C1P2R13
Description There are three possible shapes (called 'gestalten' in [dCM]) for sixth chords, abbreviated G1, G2 and G3. In connections with main triads in root position, it is advisable to use the following progressions: G1–Close, Close–G2, G2–Open, Open–G3 in case of ascending melody, and G3–Open, Open–G2, G2–Close, Close–G1 is case of descending melody.

Rule number C1P2R14
Description Covered parallels can be avoided, in case a sixth chord is followed by a chord in root position, if in the latter chord the third lies in the soprano.

B.4 Section 3 Main triads in second inversion

This section lists the harmony rules of the third section of [dCM]:

B.4.1 Strict rules

Rule number C1P3R0
Description In main triads in second inversion, the fifth has to be duplicated.
Covered by $P11$

Rule number C1P3R1
Description Cadential six-four chords come at a relatively strong beat. An exception can be made in the one-to-last measure, in case of a hemiola.
Covered by $AllRulesFiguring$, $LogFiguringRules$

Rule number C1P3R2
Description The cadential six-four chord is only possible in I_4^6–V and IV_4^6–I.
Covered by $AllRulesFiguring$, $LogFiguringRules$

Rule number C1P3R3
Description In the progression IV–I_4^6, there should be contrary motion between bass and soprano, or the soprano must have a common tone.
Covered by $AllRulesNGOV$, $LogVoiceLeadingRulesOuterVoices$

Rule number C1P3R4
Description The root and third of a cadential six-four chord have to descend stepwise. An exception can be made if both lie in a middle voice.
Covered by $AllRulesNGOV$, $AllRulesInnerVoices$
$LogLeadingRulesOuterVoices$, $LogVoiceLeadingRulesInnerVoices$

Rule number C1P3R5
Description Passing, auxiliary and arpeggiated six-four chords come at a relatively weak beat.
Covered by $AllRulesFiguring$, $LogFiguringRules$

Rule number C1P3R6
Description In progressions with passing six-four chords, the root note instead of the fifth can be duplicated in the six-four chord if the melody leaps upward a third and a fourth, or downward a fourth and a third.
Covered by $P12$

Rule number C1P3R7
Description The passing six-four chord is only possible in I–V_4^6–I^6, I^6–V_4^6–I, IV–I_4^6–IV^6 and IV^6–I_4^6–IV.
Covered by $AllRulesFiguring$, $LogFiguringRules$

Rule number C1P3R8
Description The auxiliary six-four chord is only possible in I–IV_4^6–I, V–I_4^6–V, I–V_4^6–I, I^6–V_4^6–I^6, IV–I_4^6–IV and IV^6–I_4^6–IV^6.
Covered by $AllRulesFiguring$, $LogFiguringRules$

Rule number C1P3R9
Description The arpeggiated six-four chord is preceded and followed by the same chord in root position or first inversion.
Covered by $AllRulesFiguring$, $LogFiguringRules$

B.4.2 Heuristics

Rule number C1P3R10
Description In case of a given soprano, the places where a passing six-four chord can be written should be located beforehand, and subsequently the corresponding bass notes must be written.

Rule number C1P3R11
Description Possible applications of the cadential six-four chord must be located beforehand, initially in the points of repose (half cadence and final cadence).

Rule number C1P3R12
Description The arpeggiated six-four chord must be applied on one's own initiative, e.g. in case of long notes or arpeggiation.

Appendix C

A chord progressions grammar

This appendix contains a grammar $\mathcal{G}$ for the language of permitted chord progressions, using the figuring symbols introduced in the introduction and the first three sections of [dCM]. The non-terminal symbols are capital letters from the alphabet, the terminals are the figuring symbols corresponding to main triads in all inversions. The production rules are given by $\mathcal{P}$, the start symbol is S.

$\mathcal{G} = \big(\{A, B, C, D, E, F, G, H, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z\},$
$\{\mathrm{I}, \mathrm{I}^6, \mathrm{I}^6_4, \mathrm{IV}_{(md)}, \mathrm{IV}^6_{(md)}, \mathrm{IV}^6_{4(md)}, \mathrm{IV}^{o6}, \mathrm{V}_{(o)}, \mathrm{V}^6_{(o)}, \mathrm{V}^6_4\}, \mathcal{P}, S\big)$

$\mathcal{P}$:

$S \to \mathrm{I}\ A$

$A \to \mathrm{I}\ A \mid \mathrm{I}^6\ B \mid \mathrm{I}^6_4\ C \mid \mathrm{IV}_{(md)}\ D \mid \mathrm{IV}^6\ E \mid \mathrm{IV}^6_{md}\ L \mid \mathrm{IV}^6_{4(md)}\ F \mid \mathrm{IV}^{o6}\ G \mid \mathrm{V}_{(o)}\ H \mid \mathrm{V}^6_{(o)}\ J \mid \mathrm{V}^6_4\ K$

$B \to \mathrm{I}^6\ B \mid \mathrm{I}\ A \mid \mathrm{I}^6_4\ C \mid \mathrm{IV}_{(md)}\ D \mid \mathrm{IV}^6_{(md)}\ E \mid \mathrm{IV}^6_{4(md)}\ F \mid \mathrm{IV}^{o6}\ G \mid \mathrm{V}_{(o)}\ H \mid \mathrm{V}^6_{(o)}\ J \mid \mathrm{V}^6_4\ K$

$C \to \mathrm{I}^6_4\ C \mid \mathrm{I}\ A \mid \mathrm{I}^6\ B \mid \mathrm{V}\ H$
C: current chord I^6_4, following I or I^6

$D \to \mathrm{IV}_{(md)}\ D \mid \mathrm{IV}^6\ E \mid \mathrm{IV}^6_{md}\ L \mid \mathrm{IV}^6_{4(md)}\ Z \mid \mathrm{IV}^{o6}\ G \mid \mathrm{I}\ A \mid \mathrm{I}^6\ B \mid \mathrm{I}^6_4\ M \mid \mathrm{V}_{(o)}\ P \mid \mathrm{V}^6_{(o)}\ X$

$E \to \mathrm{IV}^6\ E \mid \mathrm{IV}^6_{md}\ L \mid \mathrm{IV}_{(md)}\ D \mid \mathrm{IV}^6_{4(md)}\ Z \mid \mathrm{IV}^{o6}\ G \mid \mathrm{V}_{(o)}\ P \mid \mathrm{V}^6_{(o)}\ X \mid \mathrm{I}\ A \mid \mathrm{I}^6\ B \mid \mathrm{I}^6_4\ M$

$F \to \mathrm{IV}^6_{4(md)}\ F \mid \mathrm{I}\ A$
F: current chord $\mathrm{IV}^6_{4(md)}$, following I or I^6

$G \to \mathrm{IV}^{o6}\ G \mid \mathrm{V}\ P \mid \mathrm{I}^6_4\ M$

$H \to \mathrm{V}_{(o)}\ H \mid \mathrm{V}^6_{(o)}\ J \mid \mathrm{V}^6_4\ R \mid \mathrm{I}\ A \mid \mathrm{I}^6\ B \mid \mathrm{I}^6_4\ N \mid \mathrm{IV}^6\ O$
H: current chord $\mathrm{V}_{(o)}$, not following IV or IV–I^6_4

$J \to \mathrm{V}^6_{(o)}\ J \mid \mathrm{V}\ H \mid \mathrm{V}^6_4\ R \mid \mathrm{I}\ A \mid \mathrm{I}^6_4\ N \mid \mathrm{IV}^6\ Q$
J: current chord $\mathrm{V}^6_{(o)}$, not following a subdominant

$K \to \mathrm{V}^6_4\ K \mid \mathrm{I}\ A \mid \mathrm{I}^6\ B$
K: current chord V^6_4, following I or I^6

$L \to \mathrm{IV}^6_{md}\ L \mid \mathrm{IV}_{md}\ D \mid \mathrm{IV}^6_{4md}\ T \mid \mathrm{IV}^{o6}\ G \mid \mathrm{V}\ P \mid \mathrm{I}^6_4\ M$

$M \to \mathrm{I}^6_4\ M \mid \mathrm{V}\ P \mid \mathrm{IV}_{(md)}\ D \mid \mathrm{IV}^6\ E \mid \mathrm{IV}^6_{md}\ L \mid \mathrm{IV}^{o6}\ G$
M: current chord I^6_4, following $\mathrm{IV}_{(md)}$ or $\mathrm{IV}^6_{(md)}$

$N \to \mathrm{I}^6_4\ N \mid \mathrm{V}\ H$
N: current chord I^6_4, following $\mathrm{V}_{(o)}$ or $\mathrm{V}^6_{(o)}$, before that no subdominant

$O \to \mathrm{IV}^6\ O \mid \mathrm{V}^6_{(o)}\ J$
O: current chord IV^6, following V

$P \to \mathrm{V}_{(o)}\ P \mid \mathrm{V}^6_{(o)}\ X \mid \mathrm{V}^6_4\ U \mid \mathrm{IV}^6\ O \mid \mathrm{I}\ V \mid \mathrm{I}^6\ B \mid \mathrm{I}^6_4\ W$
P: current chord $\mathrm{V}_{(o)}$, following a subdominant

$Q \to \mathrm{IV}^6\ Q \mid \mathrm{V}_{(o)}\ H$
Q: current chord IV^6, following V^6, before that no subdominant

$R \to \mathrm{V}^6_4\ R \mid \mathrm{V}\ H \mid \mathrm{V}^6\ J$
R: current chord V^6_4, following $\mathrm{V}^{(6)}_{(o)}$, before that no subdominant

$T \to \mathrm{IV}^6_{4md}\ T \mid \mathrm{IV}^6_{md}\ L \mid \mathrm{IV}_{(md)}\ D \mid \mathrm{I}\ A \mid \mathrm{IV}^{o6}\ G$
T: current chord IV^6_{4md}

$U \to \mathrm{V}^6_4\ U \mid \mathrm{V}\ P \mid \mathrm{V}^6\ X$
U: current chord V^6_4, following $\mathrm{V}^{(6)}$, before that a subdominant

$V \to \mathrm{I}\ V \mid \mathrm{I}^6\ B \mid \mathrm{I}^6_4\ C \mid \mathrm{IV}_{(md)}\ D \mid \mathrm{IV}^6\ E \mid \mathrm{IV}^6_{md}\ L \mid \mathrm{IV}^6_{4(md)}\ F \mid \mathrm{IV}^{o6}\ G \mid \mathrm{V}_{(o)}\ H \mid \mathrm{V}^6_{(o)}\ J \mid \mathrm{V}^6_4\ K \mid \varepsilon$
V: current chord I, following subdominant–dominant

$W \to \mathrm{I}^6_4\ W \mid \mathrm{V}\ P$
W: current chord I^6_4, following $\mathrm{V}^{(6)}$, before that a subdominant

$X \to \mathrm{V}^6_{(o)}\ X \mid \mathrm{V}_{(o)}\ P \mid \mathrm{V}^6_4\ U \mid \mathrm{I}\ A \mid \mathrm{I}^6_4\ W \mid \mathrm{IV}^6\ Y$
X: current chord $\mathrm{V}^6_{(o)}$, following a subdominant

$Y \to \mathrm{IV}^6\ Y \mid \mathrm{V}_{(o)}\ P$
Y: current chord IV^6, following V^6, before that a subdominant

$Z \to \mathrm{IV}^6_{4(md)}\ Z \mid \mathrm{IV}_{(md)}\ D \mid \mathrm{IV}^6_{(md)}\ E \mid \mathrm{I}\ A$
Z: current chord $\mathrm{IV}^6_{4(md)}$, following $\mathrm{IV}^{(6)}_{(md)}$

Appendix D

Glossary

This appendix gives an elementary explanation for some of the more specialized musical terms used in this thesis. These explanations are meant to give an intuitive idea for the terms, not to establish any strict definitions.

alto The second to highest voice in a harmony exercise.

bass The lowest voice in a harmony exercise.

cadence (or final cadence) A chord progression consisting of a subdominant chord, followed by a dominant and a tonic chord. The dominant and tonic chords are in root position and the soprano ends on the root note, which is reached stepwise. All harmony exercises end with a final cadence.

caesura A caesura relates to a musical phrase like a comma to a sentence.

close spacing A spacing of chord tones in which there is no chord tone between the tenor and the alto voice, and no chord tone between the alto and soprano voice. In a close spacing, the notes in the three highest voices are placed as close to each other as possible.

dominant A chord with a tendency to steer the music towards the tonic. Its main representative is V.

enharmonically equivalent Two notes are called enharmonically equivalent if they differ in spelling but not in pitch. For instance, c-sharp and d-flat in middle-c octave are enharmonically equivalent.

half cadence A point of repose on V, often at the end of measure $4n, n \in \mathbb{N}^+$. At a half cadence, the second note of the scale is in the soprano.

harmonic rhythm A rhythmic pattern that indicates the average chord duration.

hemiola A metrical phenomenon that temporarily changes the meter by replacing two measures in $\frac{3}{4}$ (or $\frac{3}{2}$, $\frac{3}{8}$, etc.) time by one measure in $\frac{3}{2}$ (or $\frac{3}{1}$, $\frac{3}{4}$, etc.) time, or by replacing one measure in $\frac{6}{8}$ (or $\frac{6}{4}$, etc.) time by one measure in $\frac{3}{4}$ (or $\frac{3}{2}$, etc.) time. This intended meter change is not notated in the score.

figuring symbol A chord symbol that is written as a Roman numeral. I represents the triad with the first note of the scale as the root note, II represents the triad with the second note of the scale as the root note, etc.

given voice Either the soprano or bass voice of a harmony exercise, which is given in its entirety. The given voice must not be altered when completing the exercise.

inner voices The alto and tenor voice.

main triad A triad built upon either the first, fourth or fifth note of the scale (i.e. either I, IV, or V).

moll-dur (md) A term that denotes the lowering of the sixth note of the scale.

mode A term that, together with a pitch class, determines a key. In this thesis we only distinguish between the major and the minor mode.

non-given outer voice The outer voice that is not given by the exercise. In case the soprano is the given voice, the non-given outer voice is the bass, and vice versa.

open spacing A spacing of chord tones in which there is exactly one chord tone between the tenor and the alto voice, and exactly one chord tone between the alto and soprano voice.

ornamental note A note that is (to be) accompanied by a chord of which it is not a chord tone.

pitch class A class of pitches that are exactly one or more octaves apart from each other. For instance, the pitch class C contains all c's in all octaves.

root position Position of a chord of which the root note lies in the bass.

second inversion A permutation of chord tones, such that the third of the chord is in the bass. A chord in second inversion is a sixth chord.

six-four chord A chord of which the fifth lies in the bass.

sixth chord A chord of which the third lies in the bass.

soprano The highest voice in a harmony exercise. The soprano voice contains the melody.

subdominant A chord with a tendency to steer the music away from the tonic. Its main representatives are IV and II.

tenor The second to lowest voice in a harmony exercise.

tonic A chord that creates a sense of musical relaxation. Its main representative is I.

voice leading rules The harmony rules that dictate the development of a voice.

Bibliography

[Ass01] MIDI Manufacturers Association. *Complete MIDI 1.0 Detailed Specification*. 2001.

[BT94] M.I. Bellgard and C.P. Tsang. Harmonizing music the boltzmann way. *Connection Science*, 6(2-3):281–297, 1994.

[Cam94] E. Cambouropoulos. Markov chains as an aid to computer assisted composition. *Musical Praxis*, 1(1):41–52, 1994.

[Cop96] D. Cope. *Experiments in musical intelligence*. Madison, WI: A-R Editions, 1996.

[dCM] H. de Croon and H. Maas. *Functionele harmonie*. Fontys Conservatorium, Tilburg, The Netherlands.

[Dij75] E.W. Dijkstra. Ewd472: Guarded commands, non-determinacy and formal derivation of programs. *Communications of the ACM 18*, 8:453–457, 1975.

[Far90] P.F. Farrand. Method and apparatus for representing musical information, 1990. United States Patent 4960031.

[HLZ96] M. Henz, S. Lauer, and D. Zimmermann. Compoze – intention-based music composition through constraint programming. In *8th IEEE International Conference on Tools with AI*, pages 119–126, Toulouse (France), 1996. IEEE Computer Society Press.

[LJ83] F. Lerdahl and R.S. Jackendoff. *A generative theory of tonal music*. MIT Press Cambridge, 1983.

[PR95] F. Pachet and P. Roy. Mixing constraints and objects: a case study in automatic harmonization. In *TOOLS Europe '95*, pages 119–126. Prentice-Hall, 1995.

[PR98] F. Pachet and P. Roy. Formulating constraint satisfaction problems on part-whole relations: the case of automatic harmonization, journal = ECAI Workshop on Constraint techniques for artistic applications, Brighton. 1998.

[PR01] F. Pachet and P. Roy. Musical harmonization with constraints: a survey. *Constraints Journal*, 6(1):7–19, 2001.

[Sch84] B. Schottstaedt. Automatic species counterpoint. Technical Report STAN-M-19, Stanford University CCRMA, 1984.

[Ste84] M.J. Steedman. A generative grammar for jazz chord sequences. *Music Perceptions*, 2(1):52–77, 1984.

[Wid92] G. Widmer. Qualitative perception modeling and intelligent musical learning. *Computer Music Journal*, 16(2):51–68, 1992.

[WPPAT99] G. Wiggins, G. Papadopoulos, S. Phon-Amnuaisuk, and A. Tuson. Evolutionary methods for musical composition. *International Journal of Computing Anticipatory Systems*, 1999.

[Xen92] I. Xenakis. *Formalized music: thought and mathematics in music.* Pendragon Press Hillsdale NY, 1992.