A model based approach to automatic harmonization of a melody

H. V. Koops

koops@phil.uu.nl

***Abstract*—HarmTrace has shown to be an effective model to derive harmonic functions of chords in their tonal context. This article describes the addition of a chord generation and selection mechanism that investigates the abilities of the HarmTrace model as an automatic harmonizer that generates chords with a given melody. A system which is able to generate multiple harmonically well-formed chord sequences for a given melody is added to the model. From the generated sequences the best one is chosen, by means of smallest amount of parser errors. One experiment is carried out in which chord sequences for carefully selected melodies are generated. Subsequently, in a survey a panel of harmony experts are asked, based on listening to the model generated sequences, to describe their professional opinion and rating of these chord sequences. A second experiment is carried out in which a chord sequence for an selected melody is generated, and compared to the original accompaniment of that melody that is considered a ground truth. Unlike all currently existing systems, this system guarantees the generation of well-formed chord sequences, in which the constituents are automatically functionally explained with regard to their context. From the experiments, it is shown that the system is capable of generating correct chords per melody tone. However, it is also shown that harmonizing a melody with individually well-formed chord sequences from the HarmTrace grammar do not guarantee a harmonically well-sounding coherence between the sequence and the melody. This can be explained by the nature of the system, which built to analyze, rather than to create music.**

***Index Terms*—Harmony, Automatic harmonization, Haskell, Haskore, HarmTrace**

I. Introduction

EVER since the Middle Ages, when singers in monasteries began to experiment with the addition of another voice to the chant, tentative efforts are made of discovering laws that accurately describe the way multiple voices should sound together. The aspiring human mind that wants to know and understand the laws of art brought forward theories to capture this phenomenon of harmony. With time, these laws have been changing subject to fashion, politics and even ecclesiastical law. With every generation, new laws are created of the given phenomena of art, some of which are rediscovered, rewritten or changed.

The hunt for a final theory of harmony is an honest search that involves mental gymnastics of finding ways of rediscovering ourselves as a species that creates art. However, it is a false conclusion that if laws can be derived from the current musical phenomena that it will hold for future phenomena as well. Even worse is concluding that this could be used as a measure for aesthetics. For harmony theory can be seen as the collected and systemized deductions gathered by observing the practice of composers over a long time, describing what has been common practice. It is not a guaranteed way of making good music, it merely tells us how music was written in a specific timeframe of our musical history.

The practice of abstracting rules of harmony and placing them in a linguistic framework has been a part of computer science at least since the sixties of the previous century. Work by people like Jackson[1] and Winograd[2] already showed that there is a strong connection between the basic ideas of linguistics and the structure of music, by creating a grammar that describes the functional elements of a harmonic progression.

Automatic harmonization is a natural extension (and application of) harmonic analysis, and an important component in music information retrieval (MIR). Its function is to clarify principles used by composers and musicians, and to capture these in an artificial intelligence (AI) framework. Researching the mental steps involved in the process of harmonization and automating these is valuable contribution to the study of music and AI, and MIR. With the advent of ever increasing computational power, a multitude of computerized harmonization systems have been brought forward to automate the process of generating additional voices to a given melody. In addition to rule based grammar systems, statistical systems based on hidden markov models[3] and genetic algorithms[4] have been brought forward.

The rule based system HarmTrace is made in the functional programming language Haskell. Given a sequence of symbolic chord labels, HarmTrace automatically derives the harmonic function of a chord in its tonal context[5]. Among other applications, these functional annotations can be used to improve the quantification of harmonic similarity of two annotated chord sequences.[6]

Haskell[7] is a purely functional programming language with a very active research community. It is a statically typed, lazy functional programming language which allows for a very elegant and concise programming style. Instead of a computation in terms of statements that change a program state, a functional language treats a computation as the evaluation of mathematical functions and avoids state and mutable data. Functional programming has its roots in lambda calculus, a formal system developed in the 1930s to investigate function definition, function application, and recursion. Many functional programming languages can be viewed as elaborations on the lambda calculus.[8]

Because of the availability of good error-correcting parsers, and algebraic datatypes to implement context free grammar-like structures, Haskell is a good and efficient choice to implement rule-based functional harmonic anal-

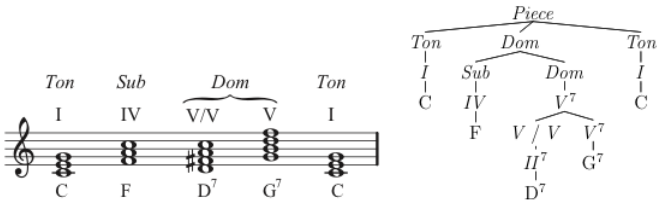


Fig. 1. A typical chord sequence and its harmonic analysis (as generated by HarmTrace) The chord labels are printed below the score, and the scale degrees and functional analysis above the score. Voice-leading is ignored for simplicity. In this case, the input was a string consisting of the key of C and a string of chords `"C:Maj,C:Maj,F:Maj,D:Sev,G:Sev,C:Maj"`. Figure taken from Magalhães and Haas (2011) [9]

ysis. The collection of Haskell datatypes in HarmTrace can be viewed as a very powerful context free grammar (CFG). The language defined by this CFG consists only of the combined values that match the structure of the datatype. HarmTrace's grammar defines the language of well-formed chord sequences, because the chords are the values, and the datatype represents the relations between the structural elements.

This article shows how a system that can complement a melody with a chord sequence can be added to HarmTrace. After generating multiple chords for melody tones, a statistical system creates a multitude of progressions. These progressions are subsequently parsed with the HarmTrace, to derive their functional structures. From these progressions, the best one is chosen by means of the least amount of errors. This harmony is combined with the input melody to create a MIDI file as an audible output. Before writing the MIDI file, an algorithm that determines the least varying inversions is used to diminish jumps in the harmony.

Simplified Haskell code examples are shown where they clarify the text, but no extended knowledge of the Haskell syntax is required to understand these. The first section of this article introduces basic music theory. The second section explains a basic harmony theory and a classic way to manually harmonize a melody. Section four explains the technical workings of the automatic HarmTrace Harmonizer that is built as an extension to HarmTrace. Section five discusses experiments conducted, and their results.

II. About music theory

THIS paragraph provides a quick introduction on music theory needed to understand the Harmtrace Harmonizer system. For a thorough understanding of music theory this paper refers to a music theory textbook[10].

One of music's basic elements is a *tone*. Its property of being quantifiable as a frequency allows the ordering of sounds on a frequency-related scale. Besides the labeling of a tone by a number representing the frequency (the number of cycles per second) in Hertz, tones can be labeled using letters. An example of this representation is Helmholtz pitch notation that uses a combination of lower and upper case letters ($A - G$ and $a - g$), and the sub- and super-prime symbols (,) and ($'$) to describe each individual tone. When a tone is described as a sign in a notation system, it is called a *note*. Notes are a combination of pitch and duration, rather than just pitch. This basic element governs melody and harmony.

In musical Set theory, pitch classes are defined by two equivalence relations. Pitches belong to the same class if they have some relation of compositional or analytical interest, for example, the octave relation[11]. The second relation is the enharmonic equivalent relation, which means that all the pitches played on the same key of an equal-tempered keyboard are in the same set. From these two equivalence relations there are just 12 pitch classes, corresponding to the notes of the chromatic scale, often numbered from 0 to 11. The choice of which pitch class to call 0 is a matter of convention or expedience. This paper sticks to the commonest conventional choice of calling C 0, in which case C♯ is 1, D is 2 and so on.

The distance between two notes, measured by their difference in pitch, is called an interval. An interval can be melodic or harmonic, in which the notes that make up that interval sound consecutively or together, respectively. In Western tonality, an interval is commonly classified as a combination between its quality (major, minor, perfect) and number (unison, second, third, etc.). The number represents the difference in the number of staff positions between two notes, and the quality derives its name the fact that some intervals come in different sizes. An example is the *third*. It is called a third because it takes up three staff positions, but can be major or minor, depending on whether it consists of three semitones, or four. The smaller interval is called a minor third, and the larger one a major third. A perfect fifth is a difference of seven semitones, and an octave spans twelve semitones. An example of the most common intervals and their qualities are listed in figure 2.

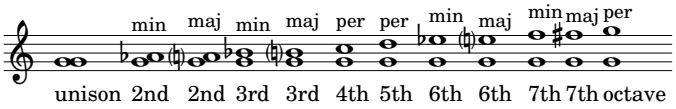


Fig. 2. Intervals and their difference in staff positions.

An interval is inverted by moving either of the notes an octave (or octaves) up or down so that both retain their pitch class.[12] For example, the inversion the interval consisting of a C with an E above it is an E with a C above it. To work this out, the C may be moved up or the E may be lowered. An interval together with its inversion yields an octave. This is because within an octave, a second inverts to a seventh, a third to a sixth, a fourth to a fifth and vice versa.

A sequence of notes in a specific ascending or descending order is called a scale. Scales are divided into categories based on the intervals they contain, being major, minor, diatonic and others. Commonly, the notes of a scale belong to a certain key. The individual notes of a scale are called scale degrees, which is a name of a note in relation to the tonic. These scale degrees can be identified in several

C major scale

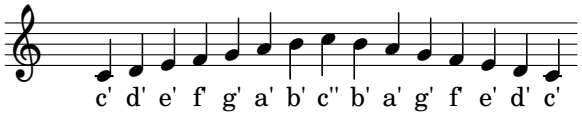


Fig. 3. C major scale with Helmholtz pitch notation

C minor scale

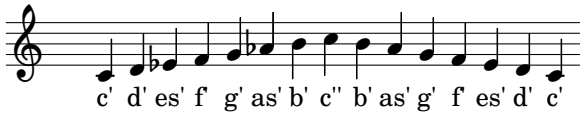


Fig. 4. C minor scale with Helmholtz pitch notation

ways: by Arabic ordinal numerals $(1,2,3,4,5,6,7)$, Roman numerals (I,II,III,IV,V,VI,VII), or by their function in the scale (tonic, supertonic, mediant, subdominant, dominant, submediant, leading note).

The combination of two or more intervals makes a chord. The most ordinary chords are called triads and are made by superposing one interval of a third and one of a fifth on a root note. The three notes the chord consist of are called root, third and fifth. Chords can be labeled in two ways: a labeling that reveals the internal structure of the chord, but tells nothing about its musical context, and a labeling that denotes the scale step upon which the chord is built. The first method, a chord name and the corresponding symbol are typically composed of one or more of the following parts:

- The chord root, for example C
- The mode of the chord, major or minor
- An added interval, like a 7th
- Altered fifth
- Additional added interval number

The other method, Roman numeral analysis, uses roman numerals in the musical analysis of chords. The Roman numeral refers to a chord built upon the corresponding scale degree. For example, the scale degree 1 in the key of C major, c, corresponds with the chord built on the first scale step, $I : Maj$. Figure 5 and 6 show all the scale degrees and their corresponding chords in major and minor. This paper uses the symbolic representation of chords as introduced by Harte et al.[13]

Functional harmony[14] is a theory of tonal harmony established by Hugo Riemann, who devised the term.[15] The theory is that each chord within a key can be reduced to one of three harmonic functions those of tonic, dominant, and subdominant. The tonic is used to affirm the key, the subdominant is used to build tension, and the dominant is used to build maximum tension. These rules are expressed in HarmTrace's grammar, where it is defined that a subdominant should always be followed by a dominant. This is illustrated in figure 8 on page 8.

The circle of fifths is a geometrical representation of relationships among the twelve pitch classes of the chromatic scale. It shows how many flats or sharps a key has, and what its relative key is (the key with the same number of

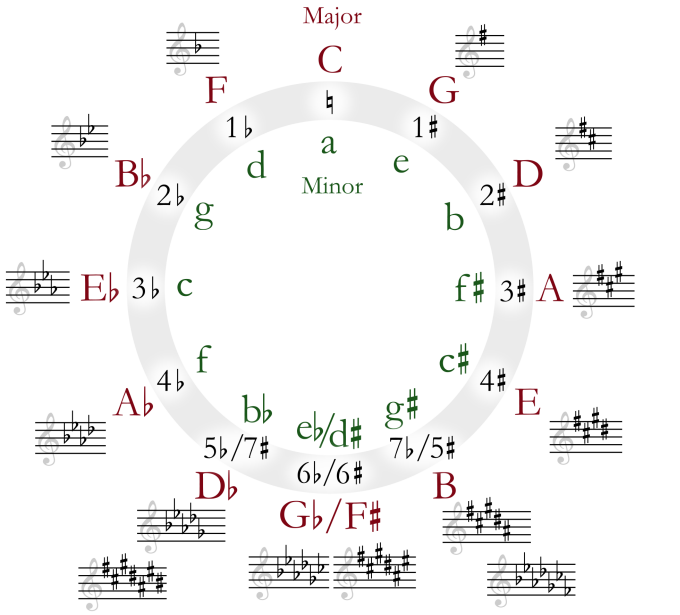


Fig. 7. The circle of fifths. [16].

accidentals, but in the opposite mode). It starts at the top of the circle with the C major key, which has no sharps or flats. Clockwise, when the notes are assumed to be in ascending order, the notes are all a fifth apart. Counterclockwise, in this same fashion, a circle of fourths is represented, since the inversion of a fifth is a fourth.

It should be taken in account that the circle of fifths names some of its notes by its enharmonic equivalents. For example, a fifth up from the note B would be called a $F\sharp$, but a fourth down from $D\flat$ would be called a $G\flat$. Although $F\sharp$ and $G\flat$ are enharmonically equivalent, that is, they represent the same sounding pitch, they represent different functions within keys and scales.

III. Harmony

> *Harmony is the study of simultaneous sounds (chords) and of how they may be joined with respect to their architectonic, melodic, and rhythmic values and their significance, their weight relative to one another.*
>
> Arnold Schönberg - *Theory of harmony[17]*

A. General harmonization

In music in general, the basic element is the tone. In harmony, the unit is the interval. Harmonization is not subject to universal laws, as all eras and even individual composers have developed their own harmony practices. The following section describes the harmonization rules that are generally applicable in western music, as it is depicted for instance by Walter Piston in his textbook *Harmony* [18], which is considered to be a classic in its field. A number of typical examples are provided to illustrate the generally applied process of harmonization.

Scale degree	1	2	3	4	5	6	7
Traditional notation	I	ii	iii	IV	V	vi	vii°
Alternative notation	I	II	III	IV	V	VI	VII
Chord symbol	I:Maj	II:min	III:min	IV:Maj	V:Maj	VI:min	VII:dim

Fig. 5. Scale degrees and chords in major

Scale degree	1	2	3	4	5	6	7
Traditional notation	i	ii°	III	iv	v	VI	vii°
Alternate notation	I	ii	iii	iv	v	vi	vii
Chord symbol	I:min	II:dim	III:Maj	IV:min	V:min	VI:Maj	VII:dim

Fig. 6. Scale degrees and chords in minor

Harmonizing a melody is the process of complementing a given melody with chords. It is characteristic of the nature of music that any melody is capable of being harmonized in more than one way. Because of this ambiguity, the process of harmonization involves a consideration of the alternatives in available chords and the reasoned choice of these chords. The process consists roughly of three consecutive processes: melody analysis, chord selection and sequence selection.

Before chords can be considered, an analysis of the melody is needed. Knowing the time signature and tonality of the melody are of vital importance of creating a functional harmony. The tonality of the melody tells us which chords can be used with the individual notes, and the time signature tells us something about the occurrence of the chords, as the strong and weak parts are treated with more and less importance with regard to chord placement. Segmenting the melody in melody parts that are based on the time signature tells us where and how many chords can be used. The structure of the melody is also a guide when choosing chords. With large melodic skips ending on weak bar structures, it is likely that the best procedure will be to use the same harmony for both notes. Likewise, if a note is sustained, a change of chords is usually not preferred.

After analyzing the melody, chords should be selected with the notes. Because a single note in the major and minor scale occurs in three common triads of the scale, a melody of n tones can be harmonized in 3^n ways when using common key chords. Quite a few of these sequences will sound unnatural and break a lot of common harmony theory rules, but they should be considered harmonizations of that melody nonetheless.

The last phase is sequence selection. Constructing a sequence out of the 3^n possibilities involves selecting chords that follow a logical, naturally sounding line, for which several heuristics can be used that are taken from general harmony theory.[18] A few of these rules for selecting harmonization chords for a melody will be presented. One example of such a rule is that a harmonization should end in a cadence, a harmonic progression of at least two chords that gives the listener a sense of concludence of a phrase. Common cadences are the authentic and the half cadence. The authentic cadence is a progression of $(V-I)$, or $(IV-V-I)$. This is considered a strong cadence as it affirms a pitch as tonic and gives the listener the feeling that the phrase is complete. The half cadence is any cadence ending on (V), preceded by any chord. Because it sounds incomplete or "suspended", the half cadence is considered a weak cadence. A few more cadences exist, and every cadence can also be described according to their voicing, something which is not further described in these examples.

A method can be used that constructs a cadence from right to left. For example, in the key of C, with a melody ending on the *tonic*, the chords $(I:Maj, IV:Maj, VI:Min)$ can be chosen for that last note, because it appears in those three chords as a tonic, a third and a fifth, respectively. If ending on an authentic or half cadence is the goal, only I and V are legal choices. In this case the cadence should be an authentic one, because I is available, and V is not, eliminating IV and VI from the possible choices. Knowing that the sequence will end on I, a perfect cadence can be made by choosing V as the chord before the last. If the note before the V allows for a IV chord, the VI can be chosen as well, creating the long form of the authentic cadence.

A few more heuristics: if notes in a bar share chords, the first chord of the bar can be sustained. If a melody begins on a strong bar part, a I is chosen as the first chord, if it begins on a weak bar part, a V is chosen as the starting chord of the sequence.

These are just a few of the many of rules the can be applied while selecting a proper harmonic sequence out of all possibilities.

IV. Related Work

QUITE a few automatic harmonization systems have already been brought forward, with different approaches to harmonization. In general, these systems can be categorized as rule based models, statistical models or genetic models. The last category is the category of hybrid models that uses concepts of more than one type of model[19]. Examples of a rule based model, two statistical models and a hybrid approached are briefly touched upon here.

Temperley and Sleator proposed the knowledge driven system The Harmonic Analyser[20] (HA) that applies a Harmonic Preference Rule System to harmonize a melody. The model has two basic tasks: it must divide the piece into chord spans, and it must choose a root label for each chord-span. After segmenting the melody, each of the possible 12 roots are assigned to each segment, which are

given a score based on the weighted sum of the rules. The four harmonic preference rules focus on preferring certain TPC (tonal pitch-class)-root relations over others, preferring chord-spans that start on strong beats of the meter, preferring roots that are close to the roots of nearby segments on the line of fifths, and preferring ornamental dissonances that are (a) closely followed by an event a step or half-step away in pitch height, and (b) metrically weak.

A data driven, statistical model that uses hidden markov models was introduced by Allan and Williams [21] Allen and Williams describe how a data set of chorale harmonizations composed by Johann Sebastian Bach can be used to train hidden markov models. A probabilistic framework allows to create a harmonization system which learns from examples, and which can compose new harmonizations. The framework of probabilistic influence allows to perform efficient inference to generate new chorale harmonizations, avoiding the computational scaling problems suffered by constraint-based harmonization systems.

Paiement, Eck and Bengio[22] propose a representation for musical chords that allow for domain knowledge inclusion in probabilistic models (e.g. major third is not likely to be played when a diminished fifth is present). A graphical model for harmonization of melodies is brought forward that considers every structural component in chord notation. The trained probabilistic models can be sampled to generate very interesting chord progressions given other polyphonic music components such as melody or root note progressions.

In a hybrid approach, Chuan and Chew proposed an Automatic Style- Specific Accompaniment (ASSA) system that generates accompaniments in a particular style to a melody given only a few training examples.[23] The system applies statistical learning on top of a music theoretic framework, therefor taking a hybrid approach. In ASSA, the relation between melodic notes and chordal harmonies is modeled as chord tone determination: if the note is part of the chord structure, then the note is classified as a chord tone; otherwise it is labelled a non-chord tone. Each melody note is represented using seventy-three attributes, including pitch, duration, metrical strength, its relation to the neighboring tones, etc. These attributes describe the functional role of each melody note in the various abstract musical structures of the song. Unlike the HA system, the preference or suitability of a certain type of note or chord is not determined by pre-programmed rules; it is learned from the training examples. The resulting classifier, a trained decision tree in ASSA, is determined by the style shared in common by the training songs.

In a comparison between of statistical and rule-based models of style-specific harmonization, Chuan[19] compared three different approaches, a rule-based model, a statistical model, and a hybrid system combining the two, for automatic style-specific harmonization in popular music. Chuan observed that the rule-based system generates the most chords within a close range of the original. With an increasing number of training examples increases, the hybrid system reports more chords identical to the original than the other systems. Although the hybrid system has the ability to generate chords that were not present in the training set, it tends to produce too many types of chords for a given song. The HMM-based system, however, produces fewer and fewer chords that are similar to the original as the size of the training set grows.

V. HarmTrace Harmonizer

THIS introduces the HarmTrace Harmonizer, an extension of the HarmTrace model that harmonizes a melody. The extension uses the analytical features of HarmTrace by keeping the number of rules to select sequences to a minimum, and uses HarmTrace's grammar for the selection of the best sequence from the generated options.

The extension of the HarmTrace system that is introduced is capable of harmonizing a melody that is given as a single track MIDI file. The other part of the extension is a knowledge based system that selects correct chord sequences from possibilities according to predefined rules. In HarmTrace, the rules of tonal harmony are formalized as a Haskell algebraic datatype. Therefore, given a sequence of chord labels from the HarmTrace Harmonizer, the harmonic function of a chord in its tonal context is automatically derived.

The handling of MIDI files is done with the Haskore[24] and the Sound.Midi libraries. The Haskore library uses a simple algebraic approach to music description and composition. With Haskore, musical objects consist of primitive notions such as notes and rests, and operations to transform musical objects to create more complex ones, such as concurrent (chords) and sequential compositions (melodies). These functions are used to create the harmonized sequences as a series of chords. The Sound.Midi library [25] can write and read MIDI files. This library is used to make an audible output of the harmonizing algorithm.

The HarmTrace Harmonizer becomes a four tier system:

1 *Generating* Harmonizing the melody by generating possible chord sequences
2 *Selecting* Apply the generated sequences to a rule based system, to eliminate unwanted and illogical sequences
3 *Parsing* Determine the best harmonization by feeding the harmonizations to HarmTrace's parser, favoring analyses with lesser errors.
4 *Post-processing* Process the sequence to find the least varying inversions of chords along a trend line, and combine these with the input melody into a MIDI output file

A. Generating

The HarmTrace Harmonizer takes a single track, monophonic MIDI track as its input. For the dissection and creation of MIDI files, the Sound.Midi library is used to extract tonal and rhythmic information. Haskore is then used to represent the MIDI file contents into

objects that represent tonal information. Because the HarmTrace and Haskore libraries use different representations for musical objects, a translator is made that can transform Haskore information into HarmTrace's datatypes and vice versa.

In this first phase, the notes of the melody of the first track of the MIDI file is transformed into a type

```
type MelodyTone  = (Root,Octave,Duration,ChordList)
```

This type represents each tone of the melody as a 4-tuple of the `Root` representation of a note from HarmTrace, an `Octave` represented by an integer which denotes the octave the `Root` note is in, the `Duration` of the note as a rational integer, and a list of chords that can be used to harmonize this note. In this phase, `ChordList` is an empty list. After the melody has been processed, the input MIDI file is analyzed and it's key signature, time signature and the transformed melody are changed into a datatype `Song`.

```
data Song = Song { key     :: MusicRep.Key
                 , timesig :: TimeSig
                 , leap    :: Integer
                 , melody  :: [(Bar,[MelodyTone])] }
```

This datatype consists of a key signature, a time signature, a leap value, and the melody. The `leap` value denotes the positions of the chords with regard to the melody. This notion is further explained in the next section. The melody in the `Song` datatype is a list of 2-tuples, in which the first is the current bar, starting at $(1, n)$ and ending with (n, n), where n is the number of bars of the melody. The second element of the 2-tuple is a list of the melody tones and possible harmonizations for that note.

A.1 Melody to pitch class

To select three possible chords with each melody tone, the tones are converted to their relative integer values. A starting value of 0 is used because Haskell uses 0 as the first element of a list. This representation should not be confused with the notion of scale degree as mentioned above. This representation uses no key information, but represents the note as absolute integer values of the chromatic scale, starting with $C = 0$, up to $B = 11$.

A.2 Creating the chord list for every melody tone

To find chords that contain a specific pitch class, possible chord structures are represented as lists of notes that are, and that are not in the chords. All chords have a specific tonal content, for example, every major chord consists of a major third and a perfect fifth on top of a root note. This can be represented as a binary list:

```
 C,  Db, D,  Eb, E,  F,  F#, G,  Ab, A,  Bb,  B
 0,  1,  2,  3,  4,  5,  6,  7,  8,  9,  10,  11
[1,  0,  0,  0,  1,  0,  0,  1,  0,  0,  0,   0 ]
```

If one wants to make a major chord on a root note D, the list is rotated 2 positions to get

```
 C,  Db, D,  Eb, E,  F,  F#, G,  Ab, A,  Bb,  B
 0,  1,  2,  3,  4,  5,  6,  7,  8,  9,  10,  11
[0,  0,   1,  0,  0,  0,  1,  0,  0,  1,  0,  0 ]
```

By representing the content of a chords as a list of ones and zeros, the content of a chord structure can be analyzed without having to list all the possible chords in every possible key.

```
shortHandToChordStructure :: Shorthand -> [Integer]
shortHandToChordStructure sh = case sh of
  --root ex. C,  Db, D,  Eb, E,  F,  F#, G,  Ab, A,  Bb,  B
  --relat.   0,  1,  2,  3,  4,  5,  6,  7,  8,  9,  10,  11
 Maj     -> [1,  0,  0,  0,  1,  0,  0,  1,  0,  0,  0,   0]
 Min     -> [1,  0,  0,  1,  0,  0,  0,  1,  0,  0,  0,   0]
 Dim     -> [1,  0,  0,  1,  0,  0,  1,  0,  0,  0,  0,   0]
```

With the integer number of the note and the representation as listed above, candidate chords of the harmonization can be found. Because a triad chord contains three notes of a scale: the root, third an fifth, the number of chords that a specific note is a part of, is 3. It is part of a chord where it is the root, a chord where is serves as third and a chord where its function is a fifth.

We can list all the possible chords of a key and mode, and convert them all to the binary representation. This creates a list of lists of binary representations, each of which are all specifically rotated, except for the one chord made on the tonic of the key. To find the possible chords for a note, a 1 must be present at position i of a binary representation, where $i =$ the chromatic integer representation of that note.

This results in three possible chords for each note of the melody, by which the `ChordList` in the `Song` datatype is expanded with chords that are allowed for those notes. A maximum of 3^n chords for n melody tones are created for a melody in a major key.

To reduce the search space, rules are applied to the possible chords to command the notion of a cadence. Since a half cadence ends in V, and an authentic cadence ends in I, chords in list of the last melody tone can be reduced to cadence-only chords. If the list contains a V or a I, it stays in the list, eliminating the other tones.

Since its uncommon to start a chord sequence on a different chord than I or V, a similar method of filtering of chords for the first melody tone can be used. Again, if the list of possible chords for the first tone in the melody contains a V or a I, they remain untouched in the list, eliminating the other tones.

A.3 Removing the leap chords

The third value of the `Song` data type is the leap value, represented as an integer number. This number shows which notes of the melody will be harmonized with a chord, and which remain unharmonized. The leap value tells us that every s melody notes there should be a chord, where s is the leap value. The `leap` value is a static value that is calculated with regard to the time signature and is roughly equivalent to the notion of strong beat parts.

```
calcLeap :: TimeSig -> Int
calcLeap (n,d) | (n == 1) = 1
               | (n == 3) = 3
               | (n == 4) = 2
               | (n == 6) = 3
               | (n == 8) = 4
```

This function gives us for the time signatures $1/d$ a leap value of 1, signatures $3/d$ and $6/d$ a leap value of 3, for signatures $4/d$ a leap value of 2 and for $8/d$ a leap value of 4, where d indicates the note value which represents one beat. We use this notion of leap to filter out chords, by eliminating those chords that do not occur on strong meter parts. Complex and mixed time signatures are ignored for now.

A.4 Giving the possible chords their probablilities

To create multiple harmonies for a given melody, a probability is calculated for each chord. This probability is calculated by the distance of the melody note to the root note of the chord in the circle of fifths. The circle of fifths is used because the tonal distances represented correspond with the human perception.

To calculate the probability of a chord for a note, the absolute pitch class (chromatic) number representation of a note is transformed into its relative pitch class (diatonic) number representation, and the note that belongs to that number is found. The scale degree notation of the chord is transformed into a letter representation, and its root note is extracted. Both of the two notes in letter format are found in the circle of fifths, from which the distance is calculated. This distance is the number of clockwise steps from the melody tone to the root note of the chord.

To convert the number representing the note in a chromatic scale into the number representing the note in a diatonic scale, a function `convergeNotes` is used.

```
convergeNotes :: Int -> [Int] -> Int -> Int -> Int
convergeNotes n (x:xs) a pos
      | (a == n)  = pos
      | otherwise = convergeNotes n xs (a+x) (pos+1)
```

This function takes an integer number, a list of integer numbers and an accumulating integer number to return an integer number. The first argument is the note represented as a number in the chromatic scale, and the list of integer numbers is a list that represents the intervals of a key. An example for a major and minor scale are listed below.

```
majRel, minRel :: [Int]
majRel     = [ 2, 2, 1, 2, 2, 2, 1 ]
minRel     = [ 2, 1, 2, 2, 1, 2, 2 ]
```

Using `majRel` in the case of a major key, and `minRel` in the case of a minor key as the second argument for the function, the function `convergeNotes` recursively increases the accumulating argument by the next element of the list, until it is equal to the value of the chromatic representation of the note. What it returns is the number of elements of a given list it must add up to become equal to the first argument of the function. This number represents the distance from the root of a diatonic scale. To convert this number into a note, all the notes of a certain key are listed in ascending order, starting from the root note of that key, and take the n^{th} element, where n is the number representing the note.

The other note needed to calculate the difference on the circle of fifths is the root note of the chord. This is relatively easy, as the `Chord ScaleDegree` (Roman numeral) representation of the chord can be transformed into a `Chord Root` (note name) representation. The Roman numeral of the scale degree is changed into its Arabic numeral equivalent minus 1, which is used as a position marker in a list of notes that the key consists of. For instance, the key of A minor consists of the notes a, b, c, d, e, f, g. The chord with the Roman numeral III has an Arabic equivalent of 3, and on position $3 - 1 = 2$ of the list of notes of the key A minor a note of c is found.

Because the representation of the circle of fifths in the HarmTrace Harmonizer is a list of datatypes that represent root notes, and one datatype cannot have two names, the representation of the circle of fifths is done via two lists. The two lists represent the same notes, except where they represent their enharmonic equivalents. `Nothing` denotes no accidentals, `Just Sh` denotes a ♯, raising the note by one semitone. `Just Fl` denotes a ♭, lowering the note by one semitone. Theoretical keys (like D♭ major) and notes with double accidentals (e.g. B♭♭ are ignored for simplicity.

```
circleOfFifths :: [Note DiatonicNatural]
circleOfFifths =  [ Note (Nothing) C
                  , Note (Nothing) G
                  , Note (Nothing) D
                  , Note (Nothing) A
                  , Note (Nothing) E
                  , Note (Nothing) B
                  , Note (Just Fl) F
                  , Note (Just Sh) C
                  , Note (Just Sh) G
                  , Note (Just Sh) D
                  , Note (Just Fl) B
                  , Note (Nothing) F ]

circleOfFifths' :: [Note DiatonicNatural]
circleOfFifths' =  [ Note (Just Sh) B
                   , Note (Nothing) G
                   , Note (Nothing) D
                   , Note (Nothing) A
                   , Note (Nothing) E
                   , Note (Nothing) B
                   , Note (Just Fl) G
                   , Note (Just Fl) D
                   , Note (Just Fl) A
                   , Note (Just Fl) E
                   , Note (Just Sh) A
                   , Note (Nothing) F ]
```

The function that calculates the distance in the circle of fifths tries to find the first note in the first list. If it fails, it tries to find its position in the second list. The same procedure is used for the second note. This way, finding the position of a note is guaranteed, because it is either in the first list, or the second list.

Now that the distance between a melody tone and the root note of a possible chord can be calculated, a probability can be given to that chord. For now, the probability

of a chord will be $1-p$, where p is the difference between the current melody tone and the root note of the chord, measures in clockwise steps on the circle of fifths.

For instance, in the key of C, a note e has three chords possible to be harmonized with: $E:Min$, $C:Maj$ and $A:Min$. The root notes of these chords are respectively e, c and a. The distance of these notes with the root note of the key, c, are 0, 4 and 1. When these numbers are individually subtracted from one, and a list of 2-tuples containing the chords and their probability is created, the list `[(I:Maj,0.6),(III:Min,1.0),(VI:Min,0.9)]` that belongs to the note e is created.

B. Selecting sequences

The process that creates chord sequences that are possible harmonizations of the melody selects chords for specific melody tones by a random selection method. The probabilities for each chord are transformed to give them their relative interval in the [0..1] interval.

When there are 2 or 3 chords that fit a melody tone, their probabilities are calculated with regard to the circle of fifths. First, a chord's distance is calculated by 1-p, where p = the number of clockwise steps in the circle of fifths from the chord's melody tone to the root of the chord. Second, this distance number is divided by the sum of the number of clockwise steps in the circle of fifths of all the chords for that note. If there is just one chord available for a note, its probability is always 1. This way, the probabilities of a chord list for a melody note always add up to 1, with each chord taking up its own relative interval. For instance, for the note e in the key of C major, the following chord list with probabilities is created: [(I:Maj,0.6),(III:Min,1.0),(VI:Min,0.9)]. To give each chord it's relative interval between 0 and 1, the list is sorted with the highest probability first. Each of the probabilities is divided by the sum of all the probabilities of the chords, in this case 2.5. This returns the relative intervals of the probabilities of the chords: [(III:Min,0.4),(VI:Min,0.36),(I:Maj,0.24)].

A uniformly distributed random number between 0 and 1 is generated. If the random number is less or equal to 0.24, chord I:Maj is chosen. If the random number is larger than 0.24, but smaller than $0.36+0.24=0.6$ then chord VI:Maj is chosen. Lastly, if the random number is larger than 0.6, chord III:Min is returned.

An integer number is given how many sequences will be generated. A recursive function repeats the steps mentioned above as many times as that number. This creates a list of lists of chords, where each list is a progression that can be used to harmonize the melody. These elements are passed through to be parsed.

C. Parsing

Because in HarmTrace tonal harmony is modeled as a Haskell datatype that closely resembles a context free grammar, the datatype will define a *language*. This language is the set of sentences that are well formed chord sequences according to the production rules of the grammar. For an in depth explanation and technical details of the encoding of concepts of harmony in Haskell, this paper refers to Magalhães and Haas (2011)[9].

1	$Piece_m$	$\rightarrow$	$Func_m^+$
2	$Func_m$	$\rightarrow$	$Ton_m \mid Dom_m$
3	Dom_m	$\rightarrow$	$Sub_m Dom_m$
4	Ton_{Maj}	$\rightarrow$	$I_{Maj} \mid I_{Maj}\ IV_{Maj}\ I_{Maj} \mid III^c_{Maj}$
5	Ton^m_{Min}	$\rightarrow$	$I^c_{Min} \mid I^c_{Min}\ IV^c_{Min}\ I^c_{Min} \mid III\flat_{Maj}$
6	Dom_m	$\rightarrow$	$V^7_m \mid V_m$
7	Dom_{Maj}	$\rightarrow$	$VII^c_{Maj} \mid VII^0_{Maj}$
8	Dom_{Min}	$\rightarrow$	$VII\flat_{Minj}$
9	Sub_m	$\rightarrow$	II^c_m
10	Sub_{Maj}	$\rightarrow$	$VI_{Maj} \mid III^c_{Maj} \mid V_{Maj}$
11	Sub_{Min}	$\rightarrow$	$VI_{Min} \mid III\flat^c_{Min} \mid V_{Min}$

where $m \in \{Maj, Min\}$ and $c \in \{\emptyset, m, 7, 0\}$

Fig. 8. Example of simplified production rules of part of the HarmTrace grammar. A valid chord sequence *Piece* is defined in rule 1 - 3 to consist of at least one and possibly more functional categories, *Func*. A functional category classifies chords as being part of a tonic (*Ton*), dominant (*Dom*), or subdominant (*Sub*) structure. Rules 3 - 11 translate the tonic, dominant, and subdominant datatypes into scale degree datatypes. Superscript m denotes whether a functional category is in major or minor. The chord class c categorizes scale degrees as one of four types of chords (denoted with superscripts) and is used to constrain the application of certain specifications. The four classes are major (no superscript), minor (m), dominant seventh (7), and diminished (0)
Figure taken from Magalhães and Haas (2011)[9]

To check the functional quality of the generated sequences, each of the sequences will be converted into a value of a Haskell datatype which captures the function of chords within the progression. These parsed pieces will then be sorted by least errors, and the best one is chosen as the eventual harmonization of the input melody.

The input sequence of the parser is a string of chords. Because of this, the chord sequences are converted to their string of characters representations. Since the method that HarmTrace uses abstracts from specific keys, the the key information should be passed along with the chord sequence. This is done by converting the key into its string representation and putting it in front of each chord list.

C.1 From `String` to `Piece`

The list-of-list of chords is used as the input for the harmtrace parser. Each sequence is transformed into a parse result by a function that takes a string of chords of a song and returns all possible parsed pieces, together with error-correction steps taken on tokenizing and musical recognition.

Because the chords that are used are formed from within the Haskell datatypes, errors on tokenizing will be zero.

C.2 Grammar

Two grammars are included in the HarmTrace package, `Pop` and `Jazz`. In this implementation, the `Jazz` grammar is used, because this is the most extensive one, featuring the most production rules. Because of this, the sequences will have a bias towards jazz harmony. However,

Magalhães and Haas (2011)[9] have shown that it can be used to analyse some classical works as well.

The production rules of the grammar define valid chord sequences. A simplified example of a part of these rules are listed in figure 8. The first rule defines that a $Piece_m$ can be created from one or more $Func_m$. These $Func_m$ are capable of being transformed into a functional entity that has a tonic or a dominant function. By rewriting each part by the rules of the grammar, a tree can be made that shows the structure of the harmony. An example of this can be found in example 6.

HarmTrace uses the error-correcting parser library uu-parsinglib[26] that uses heuristics to find a reasonable parse tree in a reasonable amount of time. We can use the amount of errors it returns after parsing as a way to sort the progressions. For most progressions, parsing will result in little or no errors. But for progression with a lot of surprising, or unexpected chords, multiple error-corrections will be necessary to create a valid analysis. The parser creates several trees for an harmonic progression, from which the best one is chosen, by selecting the shortest tree.

After parsing all the generated progressions for a melody, the trees are sorted by the least amount of errors. This is done by calculating an error measure which is the sum of the total errors of the parsing of a melody: the number of inserted and deleted progression segments. This is a good way to sort the list of parsed progressions, as the amount of errors is a measure of how "strange" the progression is to the grammar, and how many repairs it had to make to fix it. This way, the first tree that is most correct according to the grammar is chosen as the harmonic progression for the melody. When two trees have the same number of errors, the one with the least functional nodes is chosen.

D. Chord inversions

When creating a MIDI file with the melody and the chord progression, an algorithm is used that creates the least varying inversions in a sequence of chords.[24] This keeps the overall pitch of the chords as close together as possible, resulting in the least jumps of melody tones. This is done by computing the center of each chord, which is the average of all its pitches. Using these centers, the algorithm tries to keep the center as close as possible to an overall trend.

The center of the chord is calculated by taking the average of a list that represents its constituents (as pitch classes) and adding that to the root of the chord. For instance, a C major chord in root position has a pitch class root of 0, and a list of constituents $[0, 4, 7]$ (a unison, third and a fifth). The center of the chord is $((0+4+7)/2)+0 = 5.5$.

The center of the start and end chord are calculated, together with a trend line. The trend line is calculated by first calculating a steepness value of $(endCenter - beginCenter)/(p-1)$, where p is the length of the chord list. Subsequently the trend is calculated by making a list of values by calculating $(beginCenter + steep * k)$ for each k in $[0 \ldots (\text{length of the chord list - 1})]$. This yields a trend line that is used as a guideline to make chord inversions.

For each chord in the sequences an inversion is sought that is closest to the trend line value of the position of that chord. This is done by calculating the center of the inversion and comparing it to the trend line. The inversion that is closest to the trend line is chosen.

VI. Experiments and results

TWO experiments are carried out to measure the output of the HarmTrace Harmonizer.

In experiment one, a survey is carried out where three experts in harmony theory are confronted with harmonizations of melodies created by the HarmTrace Harmonizer. They are requested to give their professional opinion on the technical and creative aspect of the horizontal and vertical dimension of the harmonization. This experiment evaluates the difference between the model and human derived harmonization, and is an indicator of the naturalness of the output of the model.

In the second experiment, a comparative experiment is carried out to compare the chord sequence of a given human harmonized melody with the outcome of that same melody harmonized by the HarmTrace Harmonizer.

For both of the experiments, the melodies are taken from exercises from the second edition of Walter Pistons' Harmony. This source is chosen because it can be considered as a classical textbook on the theory of harmony.

A. Experiment one: materials

In this experiment, three experts are confronted with melodies harmonized by the HarmTrace Harmonizer. The melodies that are chosen for this survey are exercises b, c and e from chapter 7 (*Harmonization of a given part*) of Walter Pistons' *Harmony*[18]. These melodies are referred to in this paper as melody B, melody C and melody E, respectively. These melodies were chosen because of their difference in key, time signature and overall structure, to confront the system with diverse melodies from a reputable source.

The panel of experts consists of expert I, who is currently an undergraduate musicology student at the University of Utrecht. Expert MN is a composer in pop-oriented music, who studied composition and sound design. The final expert is GW, a lecturer at the Utrecht School of the Arts (HKU), at the faculty of art, media and technology (KMT).

It is interesting to see how experts from different professional musical areas derive different conclusions from listening to the pieces. Although difference in the analysis of the progression of the experts is shown, it is interesting that there is agreement in the broad sense of the appreciation and technical aspects of the pieces.

A.1 Method

For this experiment, the HarmTrace Harmonizer generated 1000 sequences for each melody, from which the best were chosen, as described in section V.

Three questions were presented that addressed the horizontal and vertical aspects of the harmonized piece. In two of these questions, experts were asked to give a rating on a scale from 1 to 5, to quantify their opinion. The question form that is used in the experiment is shown in appendix A.

Question one inquires about the technical aspect of the total progression. In addition to this question, experts are asked to explain this rating. With this question, they are also asked to give their professional opinion on the creative aspect of the progression, and to explain whether they consider the progression to be surprising, boring, or mechanical. The second question inquires about the vertical aspect of the harmonization, and asks the experts to rate every chord on a a scale of 1 to 5 (1: very correct - 2: moderately correct - 3: not correct or incorrect - 4: incorrect - 5: very incorrect), whether that chord fits the melody segment it belongs to. In addition, they are asked what chords they would replace, and in what way. The third question inquired whether the experts had anything more to say about the harmonizations in general, and in relation to each other.

A.2 Results melody B

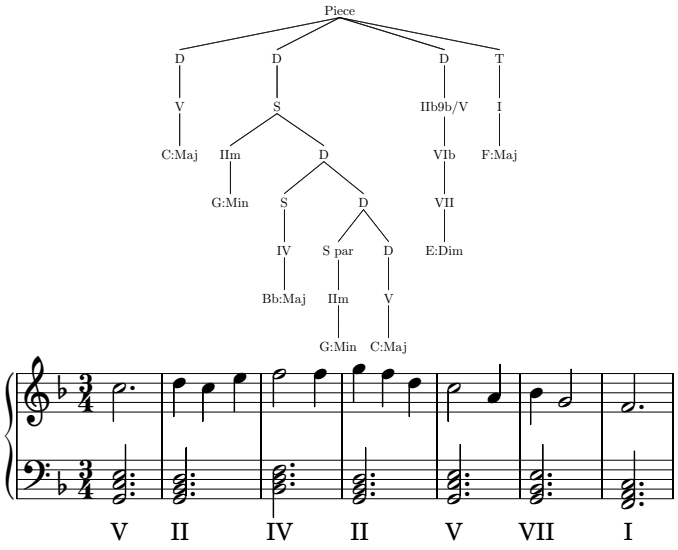


Fig. 9. Melody B in F major with harmonisation and its harmonic analysis (as generated by HarmTrace)

Expert MN notes that this harmonization makes an impression of being film music, because of the moll-dur effect, as if G minor is the key. Because it starts off with C to G major, it gives the impression that G major is the key, up until the cadence is created, where is revealed that it is in the key of F.

Chord	V	II	IV	II	V	VII	I
MN	1	1	2	1	1	1	1
I	1	2	3	1	1	1	1
GW	1	2	5	2	2	2	1
Total	3	5	10	4	4	4	3

Fig. 10. Melody B: ratings of the experts of the individual chords on a scale of 1 to 5, results of Q2 of the survey

As can be seen in figure 10, the progression of the second to third chord (II-IV) is found to be troubling. It shows from looking at the chords in the staffs that there are too many voices moving in the same direction. When moving from II to IV in F major, the algorithm to determine the shortest path did its job well, but in this case it is a bad decision. The inversion algorithm should make a different decision here, or the choice of another chord should be made.

This melody is found to be the best harmonization of the three melodies. Melody B was rated by expert MN as 1, I as 2, and GW as 3, and overall well sounding. Although not in accordance with the classical rules, the progression is regarded as a functional one, one that *works*.

A.3 Results melody C

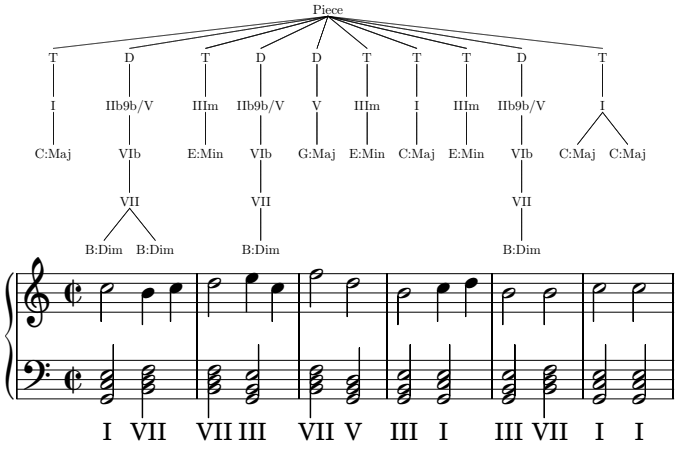


Fig. 11. Melody C in C major with harmonisation and its harmonic analysis (as generated by HarmTrace)

Melody C is the least favored melody by the experts, and was criticized with regard to the overall structure and progression.

According to the experts, the questions that were raised by the melody were not properly solved by the harmony. Although the experts agreed that individual chords fit the tones well, the progression is weak. Interestingly, the functional analysis of the progression as shown in the tree in figure 11 is one with little inner structure. An explanation for the lack of coherence in the progression could be attributed to the lack of functional components that are built out of multiple chords, or chord sequences. This way, the chords of the sequence have little functional relations with each other except for their shared key. Prudence is called for a conclusion like this, because of the low sampling size of the experiment.

Just like in melody B, the (III) chord is criticized. In this case, this can be attributed by the fact that the chord generating algorithm, which only takes the current note in account, ignores the rest of the bar. The chord does not

Chord	I	VII	VII	III	VII	V	III	I	III	VII	I	I
MN	1	3	5	3	3	1	2	3	2	1	1	1
I	1	3	1	4	1	1	1	4	2	2	2	2
GW	3	3	3	5	5	5	3	2	3	2	5	5
Total	5	9	9	12	9	7	6	9	7	6	8	8

Fig. 12. Melody C: ratings of the experts of the individual chords on a scale of 1 to 5, results of Q2 of the survey

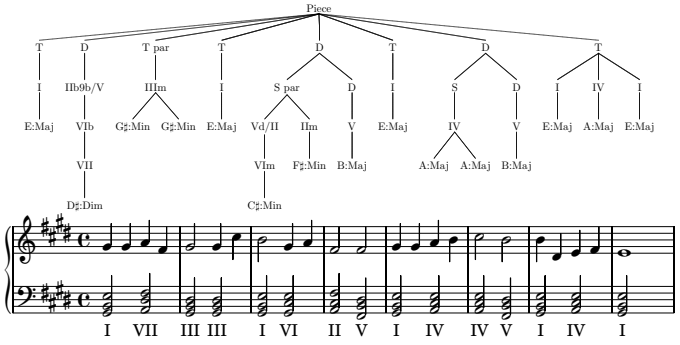


Fig. 13. Melody E in E minor with harmonisation and its harmonic analysis (as generated by HarmTrace)

solve the c, which is incorrect in the classical sense. One of the major remarks is that the center of gravity in the harmony is not in the right place. This can be attributed by the fact that the chord generation system does not take in account the position of the melody and bar that it's currently fulfilling.

Again annoyances are found that can be attributed to the inversion algorithm. Parallel and equal movements (bar 2 to 3) are found to be disturbing, and in the second parts of bar 2 and bar 4 the chords do not fit the melody. The piece starts and end on the unstable $\frac{6}{4}$ chord (the second inversion), and a doubled third of the chord is found in the melody of the first chord of the second bar. Voices are moving too little and too much in the same direction.

A.4 Results melody E

In contrast to melody C, this one is overall regarded by the experts as a good harmonization, on par with melody B.

The fourth bar sequence (II-V) is regarded as pleasant. The progression starting in meter 3 (I VI II V I) is considered to be a progression that is technically a good sequence in the classical sense. Chords that are criticized and found to not fit the melody are chord 2 of bar 2 and chord 2 of bar 3. Expert GW notes that the harmonies are not a part of a compositional idea, but are in a sense a good estimation for the moment they are used.

In figure 14, the results of the rating of the individual chords show that the most disliked chords is the (III) in the fourth bar.

Although perceived as a more pleasant progression than melody C, problems with regard to the inversions of the chords are noted. Too many equal movements in the voices of the chords are observed.

Expert	Melody B	Melody C	Melody E
MN	1	3	2
I	2	4	2
GW	3	4	3

Fig. 15. Ratings of the experts for the overall appreciation of the harmonizations created for melody B, melody C and melody E, results of Q1 of the survey

A.5 Overall results

Overall, melody B is found to be the harmonization that works best with the melody. Melody E is also relatively appreciated by the experts, but melody C is universally panned.

It it interesting to look at the functional trees of the harmonization. It shows from the figures that melody C lacks internal structure with regard to melody B and E. Great care should be taken with taking this as a cause of the quality of the harmonization, because of the low number of melodies used in this experiment.

As expert GW noted, one of the major elements that are missing are dissonant chords that are not the result of melodic jumps inside of a bar. This is the result of choosing three chords for one melody tone, where in each chord it has a consonant function as a root, third and fifth. This will never yield dissonant chords with regard to the melody, something which is regarded needed to make an interesting harmonization, as consonances are regarded to be points of arrival, rest, and resolution. Without the use of dissonance, chord sequences can sound stale and mechanical.

B. Experiment two: similarity

In this experiment, a harmonization example from Walter Piston's *Harmony*[18] is chosen as a ground truth for a harmonization from the HarmTrace Harmonizer. The harmonic structure of example 117 from *Harmony* is compared to the output of the HarmTrace Harmonizer. The manual harmonization process of this melody by Piston is extensively documented in his book. Pistons choices are compared to the choices made by the HarmTrace Harmonizer.

B.1 Method

1000 harmonizations were generated for the melody, from which the best one was chosen with accordance to the method described in section V. The first melody tone from Walter Piston's melody is doubled as two half notes, as the HarmTrace Harmonizer system cannot handle

Chord	I	VII	III	III	I	VI	II	V	I	IV	IV	V	I	IV	I
MN	1	1	1	5	2	1	1	1	1	4	2	2	1	1	1
I	1	1	2	4	2	3	2	2	2	3	2	2	3	3	2
GW	4	4	2	4	2	3	3	2	4	3	3	5	4	3	2
Total	6	6	5	9	6	7	6	5	7	10	7	9	8	7	5

Fig. 14. Melody E: ratings of the experts of the individual chords on a scale of 1 to 5, results of Q2 of the survey

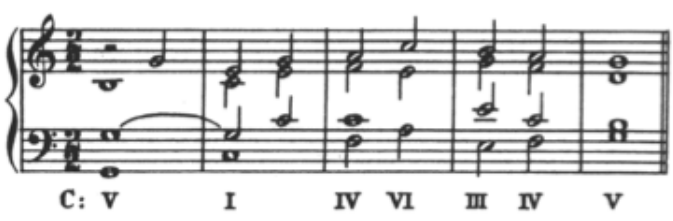


Fig. 16. Harmonization of a given part by Walter Piston as taken from *Harmony* [18]

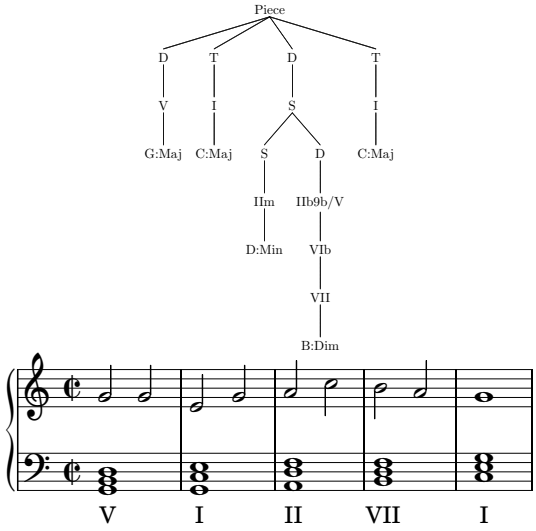


Fig. 17. Melody of Walter Piston in C major with harmonisation and its harmonic analysis (as generated by HarmTrace)

melodies that do not begin on the first beat of the first bar.

B.2 Results

The first thing that is noticed is the amount of chords. Piston chose to harmonize each melody tone with a chord, where the HarmTrace Harmonizer only created one chord per bar. This is because the leap value of a $\frac{2}{2}$ meter is 2, which means that on each two notes of the beat a chord should be made.

Both Walter Piston and the HarmTrace Harmonizer start off on the same chord. A (V) chord is chosen to open the progression. The choice of the HarmTrace Harmonizer to start on this chord can be attributed to its highest probability in the chord list of this note [(I:Maj,0.9),(V:Maj,1.0)]. Although the first note of the melody also allows for a (III) chord, this option is removed by the algorithm. This proves to be a good choice, as opening a chord sequence on a (III) chord is considered to be bad practice.

The second chord is also equivalent to Pistons choice. It affirms the key of the melody after the ambiguous start on the (V) chord. The inversion algorithm chooses the shortest path from (V) to (I), but this creates two voices moving in the same direction, which does not sound well. This also happens when the progression moves to the third chord, from (I) to (II), here moving three voices in the same direction.

Pistons solution is to choose bigger intervals for the inversions of the chords, as can be seen in the progression from (V) to (I). The inversion algorithm does not include this feature, which can lead to voice leading of the chords that exhibit too many voices moving in the same direction.

VII. Discussion and conclusion

THIS article shows how a chord generation and selection mechanism that uses the abilities of the HarmTrace model can be implemented as an automatic HarmTrace Harmonizer that generates chords with a given melody. It is shown how a system which is able to generate multiple chord sequences for a given melody is added to this model. From generated sequences created by this model the best one is selected, based on smallest amount of parser errors. Unlike currently existing systems, this system creates chord sequences from which the constituents are automatically functionally explained with regard to their tonal context.

Two experiments are carried out. In the first one chord sequences for carefully selected melodies are generated. Subsequently, a panel of harmony experts is asked based on listening to the model generated sequences, to describe their professional opinion and express it through a rating for these chord sequences.

A second experiment is carried out in which a chord sequence for a selected melody is generated, and compared to the original accompaniment of that melody.

From the comparative experiment it is shown that the system is capable of creating a harmonization that shows resemblance to the original accompaniment. From the expert opinion experiment, it is shown that the system is capable of creating reasonably well-sounding harmonizations for a given melody. However, from the expert experiment it is also shown that harmonizing a melody using the HarmTrace Harmonizer does not guarantee a harmonically well-sounding coherence between the sequence and the melody.

The process of generating chord sequences could be improved on several points. Instead of only using one melody

tone to create a list of possible chords, the melody could be segmented into sections that are governed by a chord. As a result, the probability of choosing a chord could be described as a function from all the melody tones in that segment and the way the melody is evolving to the probability of a chord. To account for dissonant chords, the chord generation process should be extended to allow chords that do not contain its current melody tone.

Another improvement could be to take into account the phrase information of the melody. Probabilities of chords could be heightened or lowered with regard to variation, strong or weak bar sections and overall melodic structure. One of HarmTrace's features is to quantify harmonic similarity of two annotated chord sequences.[27] It would be interesting to see if this method could be applied to an input melody, to derive information about the inner structure of that melody, and to look for variation and repetition. This information is highly valuable in harmonizing a melody in an interesting way, as variations and repetitions are usually reflected in a chord sequence that harmonizes a melody.

An additional suggestion for a follow up study is to investigate if the selection process of the HarmTrace Harmonizer could be stronger integrated with a HarmTrace grammar. One way to achieve this would be to use the melody to constrain parsing. Melodic information could be used in a similar way key quality information is currently used as a constraint in the grammar. As can be seen from the production rules of a grammar in figure 8 on page 8, creating a functional structure yields different results in a major or minor key. A similar approach could be used with regard to the melody, on a per-note base or per-phrase base, perhaps using similarity measures as described above.

The current inversion process uses a trend line to choose the closest inversion of a chord in a chord sequence to create the least varying harmonization. Although this works generally well, and keeps the chords close to each other, the expert experiment has shown that in some cases the least varying inversion is not the best choice. To eliminate parallel and equal voice movements, larger chord spans should be considered, extending the three possibilities of inversions with notes that are close together. Using more information of the melody, better inversion choices can be made. For example, starting and ending on the unstable $^{6}_{4}$ inversion should not be allowed in opening and cadences of a sequence, but could be used on weak bar parts. Parallel octaves that are created between the upper voice of the chord and the melody should be avoided by choosing an inversion in which the upper note harmonizes the melody tone with an interval less or greater than an octave.

The process of selecting the best harmonization for a melody uses a measure of least parser errors to determine what is the best chord sequence. From the experiments it is shown that the worst reviewed harmonization is the one with the least interesting functional tree. Although care should be taken in concluding that a tree with a complex inner structure guarantees a better harmonization, it would be interesting to see if a different measure of determining the best chord sequence could be derived from the inner structure of the functional tree. Lesser nodes on the first level of the tree forces a more complex inner structure with the same number of chords. This way, the functional complexity of the harmonization would be a measure of quality, as the number of nodes on the first level of its tree tell something about how chords can be explained with regard to their tonal context. As is shown from the tree structure of melody C, almost none of the chords share a functional identity with a neighbor. The only exception are chords that are repeated, but these do not yield an interesting progression.

VIII. Acknowledgements

I would like to thank Frans Wiering and Bas de Haas for their supervision, and Ioanna Filippidi, Michel Nienhuis, Gerard van Wolferen, Sanne Graste, Henk Koops, Barend Poot, Paul Verschuur and José Pedro Magalhães for their reviews, suggestions and expert opinions.

References

[1] R. Jackson. The computer as a 'student' of harmony. In *Tenth Congress of the International Musicological Society*, 1967.
[2] T. Winograd. Linguistics and the computer analysis of tonal harmony. *Journal of Music Theory*, 12(1):2–49, 1968.
[3] I. Simon, D. Morris, and S. Basu. Mysong: automatic accompaniment generation for vocal melodies. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 725–734. ACM, 2008.
[4] S. Phon-Amnuaisuk, A. Tuson, and G. Wiggins. Evolving musical harmonisation. In *Artificial neural nets and genetic algorithms: proceedings of the international conference in Portoroz, Slovenia*, page 229, 1999.
[5] J.P. Magalhães and W.B. de Haas. Functional modelling of musical harmony: an experience report. In *ACM SIGPLAN Notices*, volume 46, pages 156–162. ACM, 2011.
[6] W.B. De Haas, J.P. Magalhães, R.C. Veltkamp, and F. Wiering. Harmtrace: Improving harmonic similarity estimation using functional harmony analysis. In *Proceeding of the 12th International Society for Music Information Retrieval Conference*, ISMIR '11, pages 67–72, 2011.
[7] S. Jones. *Haskell 98 language and libraries: the revised report*. Cambridge Univ Pr, 2003.
[8] P. Hudak. Conception, evolution, and application of functional programming languages. *ACM Computing Surveys (CSUR)*, 21(3):359–411, 1989.
[9] W.B. de Haas, J.P. Magalhães, F. Wiering, and R.C. Veltkamp. Harmtrace: automatic functional harmonic analysis. Technical report, Technical Report UU-CS-2011-023, Department of Information and Computing Sciences, Utrecht University, 2011.
[10] S. Kostka, J.P. Clendinning, R. Ottman, and J. Phillips. *Tonal Harmony with an Introduction to Twentieth-Ce*. McGraw-Hill, 2000.
[11] J. Roeder. Pitch class. *Grove Music Online*.
[12] W. Drabkin. Inversion. *Grove Music Online*.
[13] C. Harte, M. Sandler, S. Abdallah, and E. Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *Proc. ISMIR*, pages 66–71. Citeseer, 2005.
[14] A. Whittall and A. Latham. functional harmony. *The Oxford Companion to Music*.
[15] H. Riemann. *Vereinfachte Harmonielehre; oder, die Lehre von den tonalen Funktionen der Akkorde*. Augener, 1893.
[16] JustPlainBill. Circle of fifths, `http://en.wikipedia.org/wiki/File:Circle_of_fifths_deluxe_4.svg`, January 2012.
[17] A. Schoenberg. *Theory of harmony*. Univ of California Press, 1978.
[18] W. Piston. Harmony. revised ed. *New York*, 1948.
[19] C.H. Chuan. A comparison of statistical and rule-based models for style-specific harmonization. 2011.

[20] D. Temperley. *The cognition of basic musical structures*. The MIT Press, 2004.

[21] M. Allan and C.K.I. Williams. Harmonising chorales by probabilistic inference. *Advances in Neural Information Processing Systems*, 17:25–32, 2005.

[22] J.F. Paiement, D. Eck, and S. Bengio. Probabilistic melodic harmonization. *Advances in Artificial Intelligence*, pages 218–229, 2006.

[23] C.H. Chuan and E. Chew. A hybrid system for automatic generation of style-specific accompaniment. In *4th Intl Joint Workshop on Computational Creativity*, 2007.

[24] P. Hudak, T. Makucevich, S. Gadde, and B. Whong. Haskore music notation-an algebra of music. *Journal of Functional Programming*, 6(3):465–483, 1996.

[25] H. Thielemann. *Audio processing using Haskell*. Zentrum für Technomathematik, 2004.

[26] S. Swierstra. Combinator parsing: A short tutorial. *Language Engineering and Rigorous Software Development*, pages 252–300, 2009.

[27] W.B. de Haas, M. Robine, P. Hanna, R.C. Veltkamp, and F. Wiering. Comparing harmonic similarity measures. 2010.

Appendix

A. Questions and results from the survey with experts

Survey

H.V. Koops

1 Questions

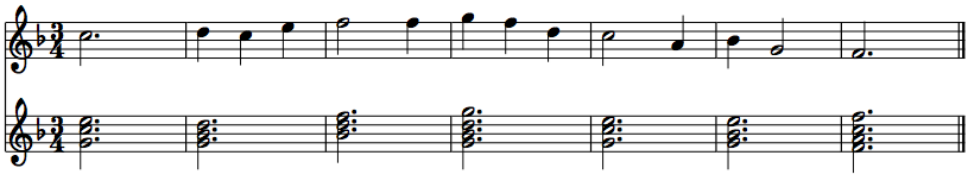
1. Please rate the technical aspect of the total progression with a number from 1 to 5, where the rating indicates whether the sequence progresses according to the general rules of harmony theory, where:
   1: very correct - 2: moderately correct - 3: not correct or incorrect - 4: incorrect - 5: very incorrect
   - a Could you explain this rating?
   - b Please tell something about the creative aspect of the total progression. (Is it surprising, boring, mechanical, etc.)
   - d Do you have any other remarks about the progression?
2. Please rate every chord and its melody segment individually with a number between 1 and 5, where the chord fits the tones :
   1: very well - 2: ok - 3: mediocre - 4: bad - 5: incomprehensible
   - a Could you clarify these ratings?
   - b Which chords would you replace, and by what chords?
3. Do you have any other remarks about the harmonization?

2 Melody B

Melody 1 in F major.

1. Rating of the total progression: ___

2. Write the numbers that go with the chord and its melody segment under the chords

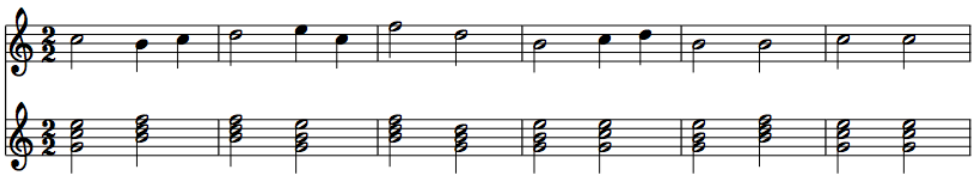


3 Melody C

Melody 1 in C major.

1. Rating of the total progression: ___

2. Write the numbers that go with the chord and its melody segment under the chords

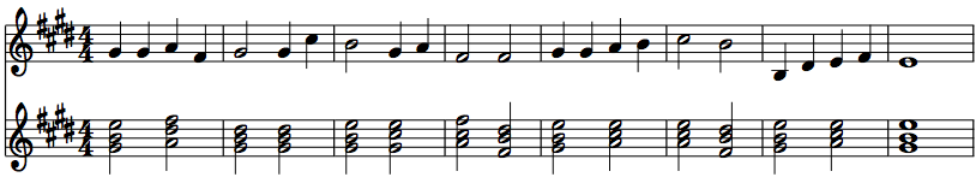


4 Melody E

Melody in E major.

1. Rating of the total progression: ___

2. Write the numbers that go with the chord and its melody segment under the chords



5 Expert I

5.1 Info

Name: Ioanna Filippidi
Info: Undergraduate in musicology, diplomas in counterpoint, musical theory, classical background.
Date: 17/1/2012
Time: 15:04 AM
Duration: 1:09:32

5.2 Transcription

5.2.1 Melody B

I: My problem is that it is in F major, and I don't see the tonic anywhere. As, in the harmonies. (About the third chord): Here you have an F (note), it would be more correct to use an F major chord to make sure that you are in F major, but it does not sound bad. It has a IV. Well it works like this. Its unusual to have this sequence in classic theory (V II IV). Especially this one. But it doesn't sound bad. The cadence is really good. It sounds like its supposed to sound. So I think it sounds rather good. Its not something that I would choose, if you have given me this melody. I wouldn't have done the same, but I guess its a matter of who does it. Also because if you have given it to a jazz player, he might have done this. Because it sounds minora (?) (inaudible) . Its weird.

V: It's based on a jazz grammar.

Q1: I: So that makes sense. Especially this progression (V I IV). But the progression is not that good, on theoretically terms, because you don't use the tonic, you use the V. So, I wil go with surprising. Well, it sounds good, but I think its ok. But this is, again, (depends) if you rate this in jazz terms or in (classical) theory term. With my classical theory, its not... If a student came up with a harmony sequence like this, I would say "Ok, but no". So I will go with OK.

Q1b: It is creative. I can go with 1. It is surprising.

Question 1c: In who's terms? Not only in one or another framework. Is it in jazz, or in classical. In jazz, I don't know. In general theory it is not correct. In normal theory it is not. In classical sense it is 3. Not correct or incorrect. Because its not bad, but not according the rules. Other remarks: it is not using the tonic one. (On the third chord) It should have used the tonic.

I have a problem with this chord. I don't know if its the fact that it goes up (II to IV), so this is what bothers me, that the whole melody goes up, and if it goes down, maybe it wouldnt have bothered me, I don't know. Maybe it's because its moving parallel. But I don't like the sound of it. On the third chord, I would prefer another.

V: You would prefer a tonic?

I: No, not necessarily, I would just prefer another way, another position (inversion). Or another chord maybe. I'm not sure. But its not what you are

waiting for. Its not a satisfying chord.

Question 2:

I: So the first is very well (good). The second is ok. The third, hmm. Three. The cadence is really good. The second chord sounds a little weird.

5.2.2 Melody C

I: Hmm. That's interesting. So, I have a problem here (III, fourth chord), because it does not solve the c. It should solve somewhere in do. It should definitely have a different chord here. Because you can really hear it, and then you don't hear the solving, in do, and it's disturbing. And I also have a problem here (III VII) I really don't like these chords. Maybe it should be the other way around. I really don't like this chord here (VII). Also, I would prefer a different chord here (VII VII), because it serves no purpose here. I don't think the harmony serves the melody.

Question 1: rate the total progression. 4. Bad. In the previous progression, it didn't make much sense in the classical sense, but this doesn't sound good. It had its moments but... The progression from here to here (VII III) is awful, from here to here was awful (III VII). So it was a bad one, I think. Maybe I'm being mean, but.

Question 1a: The harmony didn't serve the melody. It left things that you would expect, and it didn't do anything. The questions it raised weren't solved.

Question 1b: Rate the creative aspect. Trivial. That's a good word. The chords sounded good with the notes, but not with the melody (as a sequence). Maybe the following notes of the melody should be accounted for when selecting chords. The chords over two quarter notes should be for both of them. It should have something that includes them.

Question 1c. Is it a 3 or a 4? No, it's a 4.

Question 2. The first one is ok. In the second meter the sequence is bad, it needs a I in the second chord. In meter 4 to 5 the III is bad, I would have put a V there.

5.2.3 Melody E

I: This is in E major. This is good one. It's not a very good one, but an ok one. I really, really don't like this one (III on third note). It's not what you're supposed to hear.

Q1: Rating of the total progression: 2 (Ok). The general progression is good, with the exception of meter 2, where both the chords don't sound well. I like the 4th meter sequence. didn't like the same chord from meter 6 to meter 7. The cadence could be more clear with the last meter as a V, but it's not bad and it makes sense.

That's a good progression, that is why it sounds good to me, the (I VI II V I) is a good harmony.

The cadence is not much of a cadence. You should have (V I), or (V I IV I). It should follow the (V I), and you have it here but it's not a very strong one,

because you have the si (b) in both chords. You don't hear it happening. So it's not bad, but they're not clear enough.

Q1b. The creative aspect. Well, it's none. It's not trivial because it makes sense, maybe boring, but not really, I mean, I really enjoyed this one. 2. It's somewhere between surprising and boring. Not trivial, because it makes sense, and not very creative, but ok.

Q2a. In meter one, the second chord includes the melody, so it fits well. In meter two the second chord doesn't fit, just like in meter three, but not as much as in meter 2. On meter seven, maybe it should be one chord, but it's not bad.

Q2b. I would definitely replace the meter 2 chord, maybe with (I VI). It's not bad as it is, but the third should definitely be different. And on meter 5 the VI with a II maybe, and the meter seven with a V in the whole meter.

6 Expert MN

6.1 Info

Name: Michel Nienhuis.
Info: Alumnus Rock Academy, HKU Composition and Sound Design. Plays guitar/composes for several black metal bands.
Date: 19/1/2012
Time: 11:55 PM
Duration: 1:12:56

6.2 Transcription

6.2.1 Melody B

The one before the last chord to me sounds like a V, but it is a VII. It's funny, at the start, you hear a film music kind of thing. It sounds a bit mol-dur like, like G minor is the key. It sounds a bit like that Michael Jackson "Earth song" progression, (I:min IV:maj). The suggestion of this progression is created, which I really like, and eventually it is revealed that it is in the key of F. I you only hear C - G major, you think that G major is they. You can keep this up pretty long, up until you arrive at the cadence.

Q1a. I see that b flat keeps coming back. I see that the shortest inversions are chosen. I'll rate this one with a 1, because it is correct. It's also because the shortest inversion are chosen every time.

Q1b. If you look at pop music, which I was trained in, you wouldn't find a progression like this very often. It works well, but to me it sounds more classic. It sounds nice, not boring at all. In rhythmical sense it's not interesting at all, but that's besides the point. I could use this, I'm inspired to use this progression. I see it as very useful material. On a scale from 1 to 10, I would say a 7. It sounds a bit classicist or like film music.

Q2a. They all sound good. I would give them all a 1, except maybe the third one, I would replace that I (F) by a IV (B flat). It's interesting that it's

using the same chord with the variation of the melody. It would be interesting if the algorithm could use rhythmical information to choose chords.

Q2b. If I have to choose, I would maybe replace the third one, I would replace that I (F) by a IV (B flat). Maybe it's because the chord progression goes up parallel, but I'm not sure.

Q3. I miss the note f in the bar before the last bar. Two reasons, because it would repeat the rhythm of the melody, and it would include a leading tone.

6.2.2 Melody C

This one I don't like as much as melody B. The fourth chord (E minor) does not sound right. I think that there are too much VII chords in this progression. Why does it chooses the same chord twice here? (meter 1 to 2), it makes much more sense to hear a G here. I would not make this progression with this melody, I would make it much less complex.

Q1. The technical aspect, I don't know. It sticks to the key, so much is true. But I instantly think that I would make a different harmonization with this melody. I give it a 3. I think it chooses too often for a VII chord, which doesn't make sense. A diminished triad is a kind of dissonant chord, that wants to resolve into a consonant chord. That connection is really strong, of a VII to a I. It chooses to make a VII chord, then another VII chord, and then to a minor chord, which doesn't sound logical. So that a strange choice to me.

Q1b. Creatively not that strong, because better choices could be made. The fourth chord from the last should not be a E minor.

Q2. It's not logical to change to a I chord inside the bar, it's kind of mechanical.

6.2.3 Melody E

After looking at it: Oh, I'm very interested in this one. This melody has a distinct rhythmical structure. You can clearly hear that the second half of the melody is a variation on the first half. It's very good that it has chosen a I chord on the beginning of the second half of the melody. I don't know if the system has a notion of these structures, but this is has done this very well. The only thing is the fourth chord, it should have chosen another chord. It should be (I V - III VI). The G♯:minor (III) then acts like a miniature V for VI. That is what I instantly hear as more logical. The (I VI II V) is a very strong choice, which is a very common progression. The (VI II V) acts like an omen for the (I) of the second part of the melody. The choice of a VI in measure 6 is not that strong, a different chord would make a nicer progression. Maybe a I in a second inversion. But you should space the bass note an octave away from the chord to make it sound nice.

Q1a. A Two, because I had a few remarks.

Q1b. It doesn't sound very surprising. Maybe thats a good thing, because it made logical choices. The good thing is that the harmony changed when the note stayed the same. The first progression was more interesting, but maybe thats

also because of the melody. The melody is a kind of "Alle menschen werden Brüder", a really simple and classicist melody, which I'm not that interested in.

Q2. The fourth chord gets a 5 because I instantly had a remark for it.

I think melody b is the best one when it comes to harmony, then this one, and melody c comes last. It's very good that it made a half cadence for the first half for the melody. It makes me wonder how it made this choice.

7 Expert GW

7.1 Info

Name: Gerard van Wolferen.
Info: Teacher at the Utrecht School for the Arts
Date: 23/1/2012
Time: 11:55 PM
Duration: 1:12:56

7.2 Transcription

(GvW was not very talkative at the beginning of the interview)

7.2.1 Melody B

Q1 3. Not correct.

The parallel triads in root position are considered bad, and should not appear at all, including the fifth movements that follow from them.

Q1a Parallel octaves in the melody and upper voice of the harmonies. To many voices are moving in the same direction.

Q2.b The third chord could be replaced by the same chord in a different inversion, or a D or F chord.

Q3. With the exception of the third chord it follows the rules by the book.

7.2.2 Melody C

Q1. 4.

Q1a The center of gravity in the harmony is not in the right place. Disturbing parallel and equal movements (bar 2 to 3) In bar 2 part 2 the chord does not fit the melody. Also in bar 4 part 2. The piece starts and end on the unstable $^{6}_{4}$ chord. The third is doubled in the second bar, first chord. Voices are moving too little and too much in the same direction.

7.2.3 Melody E

Q1. Rated with a 3.