Musicae Scientiae
2015, Vol. 19(2) 191–213
© The Author(s) 2015
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/1029864915581131
msx.sagepub.com

A harmonic analysis network

Ruhan Alpaydin
Istanbul Bilgi University and ITU-MIAM

Guerino Mazzola
School of Music, University of Minnesota

Abstract
Harmonic analysis of symbolic music provides useful insights into music composition and performance. Automation of harmonic analysis could provide further help in making the analysis process explicit, precise, and user-configurable. The HarmoRubette component in the RUBATO music software environment was developed for such purposes. We decomposed, redesigned and extended HarmoRubette into a Harmonic Analysis Network. We describe our tonality/Harmonic Analysis Network in detail and discuss two sample analyses with the network.

Keywords
automated harmonic analysis, computing in musicology, key finding, harmonic analysis, tonality analysis, Riemann function, RUBATO

Introduction

In tonal polyphonic music, tonality is expressed vertically via chords and horizontally via chords' motion and voice-leading. In tonal monophonic music, tonality is deduced from the melodic line only. In this case, tonality, which is a high-level feature can also be considered as "implied harmony". Thus, harmonic analysis is tonality analysis, or in other words harmony is manifestation of tonality both horizontally and vertically. Automated melody harmonization, a sister problem to automated harmonic analysis (AHA), in fact has tonality analysis (on the melody) as the first step.

In music education, harmonic analysis is the foremost step in the analysis of tonal music because in these genres (baroque, classical and romantic), form is tightly coupled with harmony. Thus, it would be right to say that analysis of polyphonic tonal music is to a large extent analysis of harmony. This analysis includes labeling the chords with respect to a key, labeling cadences, labeling notes such as passing and neighboring notes, assigning functions to chords, and finding prolongational harmonies. The result of this effort are musical segments of different tonalities and a description of how musical entities function with respect to these tonalities.

Corresponding author:
Ruhan Alpaydin, Istanbul Bilgi University, Mathematics Department, Istanbul, Turkey.
Email: ruhan.ikeda@bilgi.edu.tr

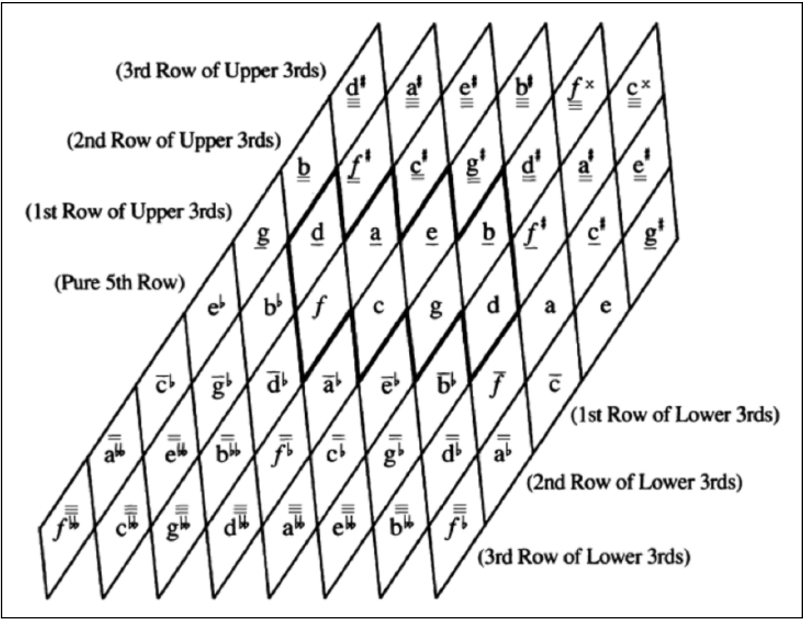


Figure 1. Riemann's visualization of tonalities' relationship.

These are the steps of a music theorist's analysis with the score in front of their eyes and the music in their ears.

The harmonic motion, created by a chords' function in a tonality, has an emotional projection: the tension that rises or falls, suspends or resolves, is perceived by listeners; a rise in tension raises an expectation for resolution, whereas a resolution is a decrease in harmonic tension. The fine balance between increase and decrease in tension is a critical component in the aesthetics of harmonic language.

Functional harmony and models of tonality

Hugo Riemann, a 19th century music theorist, established the basis for functional harmonic analysis. Riemann considered minor and major third intervals as duals of each other and he based his tonality concept on chords made out of major and minor triads, which he named as *primary triads*. His depiction of the relationship between tones and tonalities can be seen in Figure 1 (Riemann, 1992). This is an original tonality model.

Neo-Riemannian theory extended Riemann's model assuming equal temperament (and thus enharmonic equivalence) and octave equivalence. In this case, the infinite surface in Figure 1 is turned into a (two-dimensional) torus. Neo-Riemannian theory relates harmonies based on transformations that form an algebraic group (Hyer, 1995). Though Riemann himself thought about these logical connections between chords, his thinking was based on just-tuning. More recently, models of tonality have used other visual analogies (Chew, 2001; Krumhansl, 1990).

Literature on AHA

As a music theory practice, the extent and content of harmony analysis in tonal music is not precisely defined. For this reason, AHA tools also have different means and ends. A lot of harmonic analysis-related research concentrates on key-finding only, where the tonality of a music segment (monophonic or polyphonic) is found. When we consider all key-finding research, some consider modulation as well, whereas others presume mono-key segments. Since segmentation and harmonic analysis are not separate processes, key-finding in segmented/mono-key music is an easier problem.

AHA has been an important area of research with the growing amount of online digital audio and music. AHA can be useful in a number of ways: listening to and enjoying music is an indispensable activity for almost everybody. Therefore, scientific effort that clarifies music perception and cognition may benefit fields such as cognitive science, neural science and even linguistics.

From a music theoretical perspective, AHA's goal is to turn harmonic analysis into a well-defined, step-by-step process, i.e. an algorithm. With such an algorithm music can be analyzed quickly, precisely and objectively. Consequently, huge amounts of musical data can be analyzed. A metric for music similarity based on AHA is possible. Music can be encoded and compressed using its harmonic structure. Harmonic style analysis of a composer, a genre or a period becomes possible based on AHA findings. AHA can be used for teaching purposes as well (Taube, 1991; Taube & Burnson, 2009).

Grammar-based—hierarchical versus probabilistic—linear approaches. There have been two different approaches to AHA. One approach considers harmonic syntax similar to linguistic syntax. Thus, analysis of harmony is considered to be equivalent to parsing a musical string (of chords), which is derived from a grammar. In this case, a harmonic grammar is a finite set of rules which defines infinitely many strings, i.e. the harmonic language. Defining infinitely many strings using finitely many rules is possible via recursion.

The linguistic approach to AHA has two cognitive justifications.

- Tonal harmony has a syntax with recurring patterns. For example, consider the predominant → dominant → tonic motion that is the underlying harmonic motion in tonal music in formal units such as *period* and *sentence*. For larger structures, when we consider sonata form, it has a hierarchical structure with recurrences and therefore one may characterize it as a grammar.
- Research about the processing of language and processing of music in the brain found out that areas for language processing and music processing overlap (Patel, 2008). However, it is not known whether this intersection corresponds exactly to abstract-symbolic processing. For example, in the cases of brain damage leading to amusia or aphasia, the other capacity remains intact (Patel, 2008). More recent research proves that the role of acoustical factors, i.e. tonal relationships resulting from pitches of chords in the short-term memory outweighs symbolic manipulation in recognition of tonal harmony (Bigand, Delbé, Poulin-Charronnat, Leman & Tillman, 2014).

The second approach to AHA uses probabilistic models, such as hidden Markov models (HMMs) and neural networks (NNs). A HMM defines recurrent patterns through a finite-state automaton where transitions have probabilities. A NN mimics the learning mechanism in the brain. Through training, a high number of well-connected neurons learn to classify input. Certain grammatical features can also be captured by HMM and NN models. Context-sensitivity is represented via HMMs and hierarchical structures can be represented via hierarchical HMMs. Due to auditory short-term memory's limited capacity in music perception, a linear, "flat" approach in the case of HMM has been successful in capturing regularities and patterns in musical structure.

One important problem to overcome in "parsing" harmony is prolongation. Prolongation corresponds to a *hierarchy* of harmonic functions. That a chord's function can be prolonged means that (Caplin, 1998):

- the chord has structural relevance;
- the chords over which this function is prolonged has a surface function (which is embellishing) and a deeper-structural function which is the function of the chord that is prolonged.

Thus, prolongation induces a hierarchy (of chords and functions). This understanding is central to Schenker's theory. In Schenkerian analysis, "the" structure in the tonal music is found through prolongational analysis.

Grammar-based—hierarchical approaches. Literature about automated harmony analysis started with linguistic-based mechanisms. The first such effort, Winograd's parser, is based on a formal grammar that describes a subset of simple tonal music (Winograd, 1968). The input to the parser is a sequence of chords that are prepared from the music's score by a human. The program is a two-pass parser, which in the first pass labels chords and in the second labels chords' functions. The program deals with simple textures only.

The most profound study in this direction has been the Generative Theory of Tonal Music (GTTM) which is a collaborative work of a composer and a linguist (Lerdahl & Jackendorff, 1983). GTTM is grammar-based. There are three kind of rules in the grammar they define: well-formedness rules, preference rules and transformational rules. These rules work on meter, prolongation and grouping; and using them, a hierarchical music structure is derived. However, the rules are not deterministic as in the case of formal grammar rules, leading to many possible/preferable analyses of a given piece. The authors underline the fact that these rules are used for structural analysis of music and not to be used to generate music. Recently, GTTM was implemented as software (Hamanaka, Hirata, & Tojo, 2005).

Tonal pitch space (TPS) is an extension of GTTM where extensive analytical techniques and metrics for analysis of harmony are introduced: for example, a metric for harmonic tension between chords, differentiating sequential from structural tension (Lerdahl, 2001; Lerdahl & Krumhansl, 2007).

Another approach to automated tonal analysis has used chord templates and rule-based inference mechanisms. The Automatic Tonal Analysis of Music Theory Workbench project (Taube, 1991) tries to simulate pen-and-paper analysis of a musician. Aimed at analyzing four-part chorales, which is an important part of music theory education, the system identifies triad/sevenths, inversions, and non-harmonic tones. Parsing the score results in finding both horizontal and vertical units of interest, called "sonorities". Higher structural units are deduced from such sonorities via user-defined analytical "models" such as cadential motion. Continuation of this effort lead to the implementation of Chorale Composer that checks students' four-part writing for harmonic and voice-leading anomalies (Taube & Burnson, 2009). It uses 371 Bach chorales in its database as a reference and can also generate homework for students.

The most recent work we encountered in our literature survey is HarmTrace, a tool which uses a context-free grammar for functional analysis of tonal music. HarmTrace parses and identifies the harmonic function of chords (Bas De Haas, Magalhães, Wiering, & Veltkamp, 2011). The input to HarmTrace is again well-defined chords such as Asus4 or Dminmaj7; the output is the parse tree as in the case of a compiler's front end. HarmTrace is a work which tries to accomplish exactly what our Harmonic Analysis Network tries to achieve but in a totally different way.

HarmTrace cannot handle modulation but only change of mode. Second, it cannot parse chords made up of any group of notes as in the case of our approach. Finally, our network allows users to play with transition parameters from chord to chord, which results in different analyses.

Probabilistic—linear approaches. In AHA, statistical models and methodologies are used as well.

Krumhansl made experiments with experienced listeners to figure out the fit of each pitch-class (in the 12-tone equal tempered scale) to the given key. The outcome are key-profiles (Krumhansl, 1990). For the C major key for example, C contributed most, followed by G and E, all defining the tonic triad. Krumhansl's work restates a well-known observation but also provides exact data, turning every pitch's contribution to an exact number, forming the key-profile. (Lerdahl's basic diatonic space model in TPS aligns with Krumhansl's tone-profile (Lerdahl, 2001; Lerdahl & Krumhansl, 2007)). The Krumhansl–Schmuckler (K-S) algorithm uses these key profiles for key-finding. The algorithm extracts a vector for a music segment for every pitch proportional to its duration, and finds the correlation of this vector with 24 key-profiles (12 major and 12 minor). The key-profile with highest correlation is the key of the segment (Krumhansl, 1990). The algorithm cannot handle cases where a key may be perceived by a human, due to presence of structural chords, but cannot be deduced from such a statistical analysis based on pitch-class content only.

Temperley reconsidered the K-S key-finding algorithm as a Bayesian probabilistic model (Temperley, 2002). Probabilistic models are being used in speech recognition for phone-word mapping, where phone is the smallest atomic unit in a phoneme. Temperley considered phone-word relationship analogous to pitch-key, the phone/pitch being the surface(appearance) of the structure (word/key). His key-finding algorithm led to Melisma software suite, a general music analysis suite created by Temperley and Sleator. Key-finder in Melisma, after finding roots of chords, gives functional labels to chords in Roman numeral form (Sleator & Temperley, 2003). Melisma does the harmonic analysis according to a set of preference rules, an idea that was set forth in GTTM.

Aim of our work

In this paper, we present an AHA system based on a tonality model that considers chords being embedded in minimal chains of minor 3rd and major 3rd intervals. Minor 3rd and major 3rd intervals have been considered as building blocks of chords since Rameau considered triads and sevenths to be the basis of all harmony (Christensen, 2006, pp. 759–760). We will consider functional analysis of tonal music as an optimization problem on paths formed by such third chains.

The resultant harmonic analysis is an optimum path, based on the sequence of chains that make music and a set of user-defined parameters used to evaluate tension between consecutive third chains. The implementation is a network of four rubettes in the RUBATO environment: HarmonicAnalysisModel, ChordSequencer, HarmonicWeight and HarmonicPath. These four rubettes together with the MidiFileIn rubette make a network for harmonic analysis. We will present an example analysis and discuss results.

HarmoRubette's underlying harmonic domains

HarmoRubette was originally developed using Objective C as a component of RUBATO (Mozzola, 2002, pp. 819–823). We have redesigned and extended HarmoRubette as a sequence of rubettes in the Java-based Rubato Composer environment (Milmeister, 2009).

The Rubato Composer environment for composition, analysis, and performance is based on representing musical objects as mathematical structures: such objects are called denotators which are points in general spaces, called forms. They are recursively built via the use of category theory concepts such as limits, colimits, and powersets, and are based upon mathematical modules over general (not necessarily commutative) rings (Milmeister, 2009). Denotators vastly generalize the XML data format. Rubato's functional components that extend Rubato are called rubettes. A rubette may perform a particularly well-defined musical task communicating with other rubettes, using input and/or output, both strictly in denotator format. Rubato's functionalities are extensible by adding new rubettes as plug-in units (Milmeister, 2009).

The set of rubettes that we developed in the Rubato Composer environment can be used by music theorists for analysis and stylistic experiments and also by composers for creative purposes that have an analytical foundation. In the design of the analytical procedure, we tried to have a cognitive justification for every step. We have also tried to parameterize every step as much as possible, so that the user can have full control over the analytical procedure.

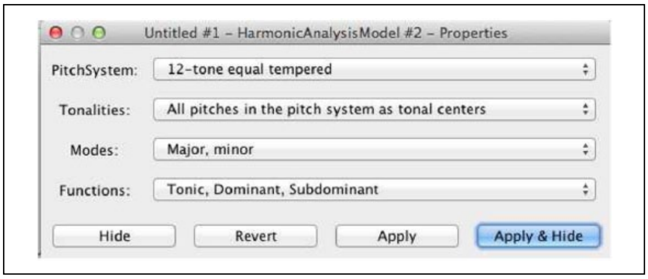
In the current version of Rubato Composer,[1] the Harmonic Analysis Network uses 12 pitch classes of the 12-tone equal tempered chromatic scale $PC = \mathbb{Z}_{12}$ as possible tonal centers; at this stage, there is no other scale system offered by the HarmonicAnalysisModel rubette. The number of modes (Major (Ionian), Minor (Aeolian), Dorian, Phrygian, etc.), and functions in the analysis can be specified by the user through the HarmonicAnalysisModel rubette. The analysis process assigns to each chord Ch a numerical weight for every possible Riemann function in the twelve tonalities and chosen modes. These Riemann functions are classical Riemann theory functions: tonic (T), dominant (D), and subdominant (S), but the set can be extended to include all diatonic functions through the RiemannAnalysisModel rubette as described in the "HarmonicAnalysisModel rubette" section. The evaluation of a chord's weight for all possible tonalities t, modes m, and functions f is represented as a three-dimensional $12 \times num_{modes} \times num_{functions}$ Riemann matrix $Weight(Ch)$. For each triple position (t, m, f) of the Riemann matrix, we have a non-negative real-valued weight coefficient $Weight(Ch)_{t,m,f} \in \mathbb{R}$. This enables a fuzzy Riemann logic, replacing the classical YES/NO choice for harmonic functions. Using the weights in Riemann matrices, a harmonic path through matrices is found under local and global constraints for chord progressions. These constraints are parameterized and can be controlled by the user.

Harmonic function defines Riemann's tonality concept

We first describe formally the musical entities used in our harmonic analysis.

A *chord* is a set $Ch \subset PC$ having the same perceived onset. Call $CHORD$ the set of all chords. This means that the chord's pitch classes can stem from pitches with different onsets, as it happens in arpeggiated chords for example, but the interpretation of these notes attributes them by definition a common onset $o \in \mathbb{R}$. This common onset is not part of the chord's identity in the sense that the common onset would be acknowledged by an ordered pair (Ch, o).

In Riemannian theory, the *harmonic meaning of a chord* is defined by the function it has within a given tonality. Thus, $F - A - C$ triad $\{5, 9, 0\}$ (with the usual identification of $C = 0$, $C\sharp = 1$, $D = 2, \ldots, B = 11$) has a Subdominant(S) function in C major while it has a Tonic(T) function in F major. Together with the third function, Dominant (D), these values define the set $F = \{T, D, S\}$ of Riemannian functions. It was Riemann's program to *define* a tonality X by the functions any chord (any set of pitch classes) Ch takes under X (Dahlhaus, 1966). This is achieved through assigning minimal third chain(s) to the chord. A complete list of third chain classes is


Untitled #1 – HarmonicAnalysisModel #2 – Properties

PitchSystem: 12-tone equal tempered

Tonalities: All pitches in the pitch system as tonal centers

Modes: Major, minor

Functions: Tonic, Dominant, Subdominant

Figure 2. Properties dialog.

in the Topos of Music book (Mazzola, 2002, pp. 1175–1181). Mathematically speaking, Riemann's program reads as the definition of a function $X : CHORD \rightarrow F$

In our model of Riemannian harmony, tonality is represented by two components: (a) a *scale*, i.e. a subset $S \subset PC$, and (b) a mode for S, which means the selection of a *tonic* pitch class $t \in S$. But it is important that this data $X = (S, t)$ does not define the function $X : CHORD \rightarrow F$, it is only a technical identification in our context. For example, the $B\flat$ major scale is $B\flat = \{10, 0, 2, 3, 5, 7, 9\}$ and the G major scale is $G = \{7, 9, 11, 0, 2, 4, 6\}$, while the G major tonality is represented by the pair $(G, 7)$. The G minor tonality can be represented by $(B\flat, 7)$ (Aeolian mode) or by the $E\flat$ major scale $E\flat = \{3, 5, 7, 8, 10, 0, 2\}$ and tonic 7 (Phrygian mode). Of course, in general contexts of musical cultures, very different scales can be applied, and Riemann's program would need to be extended to such different contexts. Such a possible extension is maqam scales which are made out of a pitch system of 30 pitch classes but not equally tempered as in the case of the common quarter tone scale.

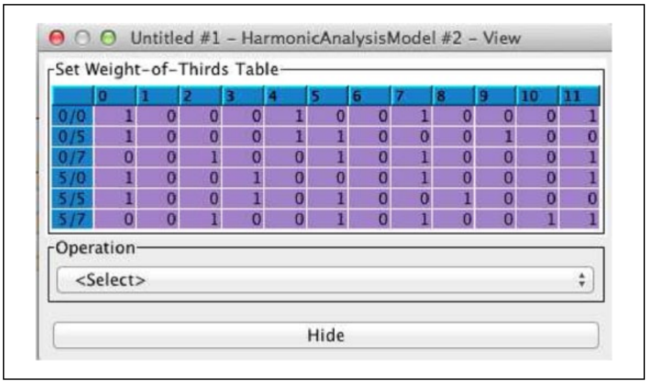
It is a crucial point of our present implementation to define functions in Riemann's unfinished plan, namely being defined for any chord, not only for major, minor or diminished triads and similar simple chords.

Four rubettes: HarmonicAnalysisModel, ChordSequencer, HarmonicWeight and HarmonicPath

Our Harmonic Analysis Network consists of four rubettes: HarmonicAnalysisModel, ChordSequencer, HarmonicWeight and HarmonicPath. These rubettes can be downloaded[2] and run in Rubato as plug-in components. We divided harmonic analysis into these four rubettes following a Unix and object-oriented programming philosophy. Each rubette's task is simple: each rubette can be controlled separately through the properties interface (playing the role of the rubette's preferences). The direction of information flow in the network formed by these rubettes is visualized by a GUI in the classical visual programming style conceived by Goldberg.

HarmonicAnalysisModel rubette

HarmonicAnalysisModel is the rubette that controls the Harmonic Analysis Network. As seen in Figure 2, the user specifies the pitch system, tonalities, functions and modes. The output of this rubette are the dimensions of the Riemann matrix to be used in the harmonic analysis. The matrix can be seen through the view dialog as in Figure 3. The columns are the tonalities that are specified and the rows are *modes* × *functions*. In the figure, rownames are Modes = {0,5}


Untitled #1 – HarmonicAnalysisModel #2 – View

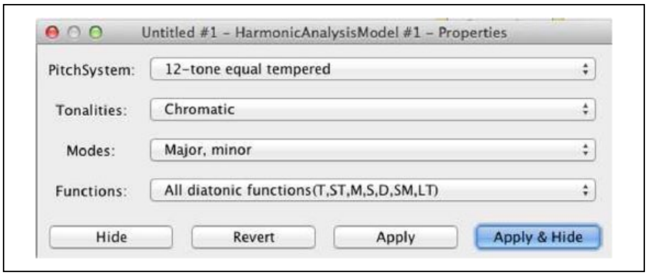
Set Weight-of-Thirds Table

	0	1	2	3	4	5	6	7	8	9	10	11
0/0	1	0	0	0	1	0	0	1	0	0	0	1
0/5	1	0	0	0	1	1	0	0	0	1	0	0
0/7	0	0	1	0	0	1	0	1	0	0	0	1
5/0	1	0	0	1	0	0	0	1	0	0	0	1
5/5	1	0	0	1	0	1	0	0	1	0	0	0
5/7	0	0	1	0	0	1	0	1	0	0	1	1

Operation

<Select>

Figure 3. View dialog.


Untitled #1 – HarmonicAnalysisModel #1 – Properties

PitchSystem: 12-tone equal tempered

Tonalities: Chromatic

Modes: Major, minor

Functions: All diatonic functions(T,ST,M,S,D,SM,LT)

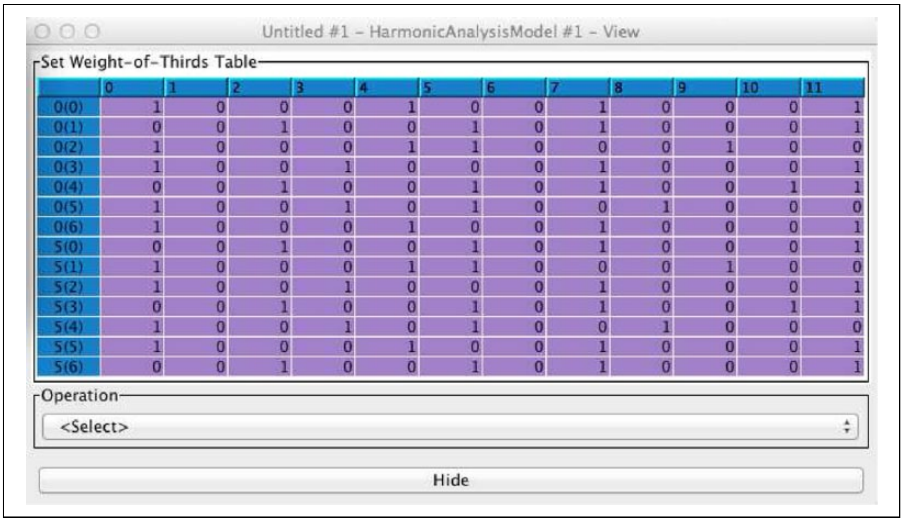
Figure 4. Properties dialog.

where 0 is for major mode and 5 is for minor mode (Minor scale starts with the 5th pitch of the major scale); functions = {0,5,7} where 0 is Tonic, 5 is subdominant and 7 is dominant function. Through the view functionality, not only the Riemann matrix size is seen, but also weight-of-thirds table's values can be set. This weight-of-thirds table will be used during evaluation of Riemann matrix values for each chord in the HarmonicWeightRubette. The weight-of-thirds table can be edited by the user, saved as a file or loaded from a file. The use of the weight-of-thirds table is discussed in the "HarmonicWeight rubette" section. Default settings assign the tonic triad with equal weight for each pitch class in the key. For example, C major row has a weight of 1 for pitch classes C, E, G, B and 0 for all other pitch classes. One may like to change this so that C as the tonic has higher weight than G which has higher weight than E and assign B the least weight. Another useful set of values would be Krumhansl's key-profile values.

In Figures 4 and 5 another setting for the HarmonicAnalysisModel rubette can be seen. In that setting, not only T, S and D, but all seven diatonic functions are chosen.

PitchSystem. A pitch system with non-zero period oct is a set S of pitches that is oct-periodic. That is to say, for each $s \in S$ it contains the orbit $s + \mathbb{Z}.oct$. A pitch system is the set of orbits $S/\mathbb{Z}.oct$ which we denote by S/oct. For example, all keys on a piano comprise a pitch system and the period is the octave. All of the white keys on the piano are another pitch system with the same period, the octave.

Currently, the only possible PitchSystem specification in the properties of the rubette is the 12-tone equal tempered scale (usually encoded by keynums in MIDI).



Set Weight-of-Thirds Table

	0	1	2	3	4	5	6	7	8	9	10	11
0(0)	1	0	0	0	1	0	0	1	0	0	0	1
0(1)	0	0	1	0	0	1	0	1	0	0	0	1
0(2)	1	0	0	0	1	1	0	0	0	1	0	0
0(3)	1	0	0	1	0	0	0	1	0	0	0	1
0(4)	0	0	1	0	0	1	0	1	0	0	1	1
0(5)	1	0	0	1	0	1	0	0	1	0	0	0
0(6)	1	0	0	0	1	0	0	1	0	0	0	1
5(0)	0	0	1	0	0	1	0	1	0	0	0	1
5(1)	1	0	0	0	1	1	0	0	0	1	0	0
5(2)	1	0	0	1	0	0	0	1	0	0	0	1
5(3)	0	0	1	0	0	1	0	1	0	0	1	1
5(4)	1	0	0	1	0	1	0	0	1	0	0	0
5(5)	1	0	0	0	1	0	0	1	0	0	0	1
5(6)	0	0	1	0	0	1	0	1	0	0	0	1

Operation

<Select>

Figure 5. View dialog.

For different pitch systems, modes and functions will be defined accordingly. For scales that use microtones, HarmonicAnalysisModelRubette should also be fed into MidiFileInRubette whose output will include Pitchbend events to represent the microtones. At this stage, this has not been implemented in the suite.

Scale. A scale in a pitch system (S, oct) is a finite non-empty subset X of S/oct. In other words, a set of pitches that is oct-periodic and have a finite set of orbits. Pentatonic scales, whole-tone scales, quarter-tone scales are examples of such scales.

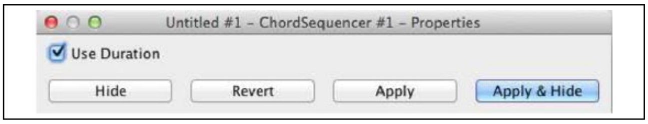
Mode. A mode is a pair (X, f) of a scale X and an element which is the tonic (or final) f of X. One may also add a second pitch class that is the tenor and denote the mode as (X, f, t) where t is the tenor. Major mode is $(\{0, 2, 4, 5, 7, 9, 11\}, 0)$, Phrygian mode is $(\{0, 2, 4, 5, 7, 9, 11\}, 4)$, etc.

In HarmonicAnalysisModelRubette, modes are represented as a sequence of intervals, starting from the tonic. Right now, there are two possible choices for Mode selection: "Major, minor", "all Greek modes".

Function and tonality. Tonality can be considered as "centricity of a tone" such as C (the pitch class C); centricity of a chord such as a C major triad; or centricity of a scale and mode, such as A minor. We use the last definition.

Function is the way a chord works with respect to a tonality. In the HarmonicAnalysisModel rubette, functions are represented as distance from tonic, thus T (tonic) is 0, D is 7, S is 5, LT (leading tone) is 11, etc. For a major scale in the 12-tone equal-tempered system, all possible function representations are $\{0, 2, 4, 5, 7, 9, 11\}$ which is the major mode itself. If $\{T, S, D\}$ functions are of interest only, then the functions are represented by $\{0, 5, 7\}$. Thus, a numerical function set specification is a subset of the mode specification. Functions are not degrees in Riemann's theory, although can be represented by degrees.

In the HarmonicAnalysisModel rubette there are two possible choices to select functions: "Tonic, Dominant, Subdominant (T, S, D)" and "All diatonic functions (T, ST, M, S, D, SM, LT)".


Untitled #1 – ChordSequencer #1 – Properties

☑ Use Duration

Figure 6. Use-duration checkbox in the properties.

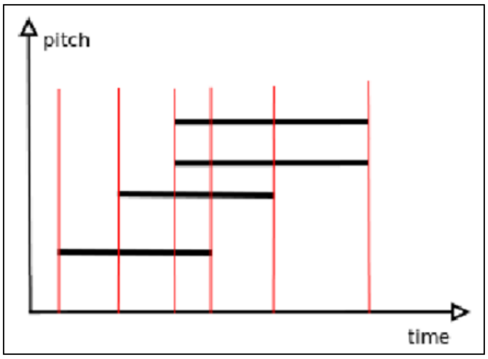


Figure 7. Four pitches in piano-roll notation. This score results in five chords if "use-duration" button is turned on by the user. This takes into account overlap in duration. If "use-duration" is off, only notes with identical onset are considered as a chord. In the second case, when "use-duration" is off, there are three chords.

ChordSequencer rubette

The ChordSequencer rubette converts a MIDI score file to a chord sequence, listing the chords for the user. This list can later be edited by the user according to individual preferences. The software creates a first list following a choice by the user regarding the role of durations to define chords. The user can turn on the use-duration property as seen in Figure 6.

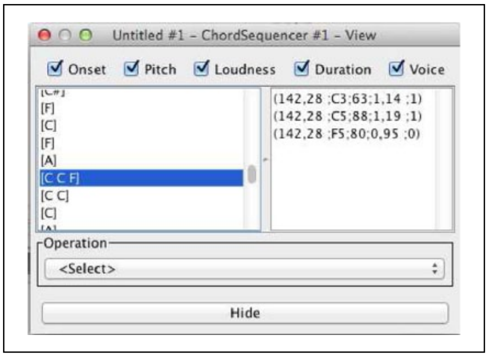
For example, the four pitches in Figure 7 (where four pitches with three different onsets overlap) would result in five chords: the first is a single note, the second is an interval, the third has four notes, the fourth has three notes, and the fifth is again an interval. If the use-duration property is off, only notes with identical onset are considered to build chords. We get three chords in our example (the first two being chords containing a single note and the last one being a chord with two notes).

Once chords are listed, as shown in Figure 8, the user is allowed to synthesize a chord from notes that do not overlap at all. Two examples in tonal music are Alberti bass and arpeggio. Both are harmonically perceived as a single chord, although notes may not overlap. This grouping also allows noncontiguous intervals be selected. Thus, the user can overwrite the rubette's automatic grouping of notes as chords, see Figure 9.

The new chord sequence can be seen in the chord list as shown in Figure 10. The duration of the notes, as well as the onsets, have changed. The new chord sequence can be used for further analysis after being saved as a MIDI file.

HarmonicWeight rubette

Once all chords are extracted from the score by the ChordSequencer rubette, the HarmonicWeight rubette constructs a Riemann matrix for each chord. The Riemann matrix is basically

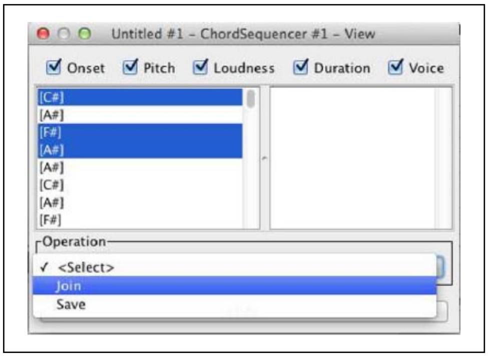


☑ Onset ☑ Pitch ☑ Loudness ☑ Duration ☑ Voice

[F]
[C]
[F]
[A]
[C C F]
[C C]
[C]

(142,28 ;C3;63;1,14 ;1)
(142,28 ;C5;88;1,19 ;1)
(142,28 ;F5;80;0,95 ;0)

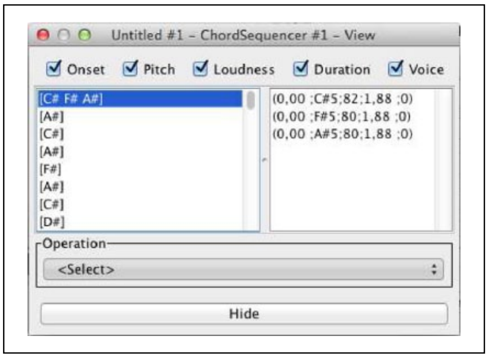
Figure 8. List of chords in the MIDI file.



☑ Onset ☑ Pitch ☑ Loudness ☑ Duration ☑ Voice

[C#]
[A#]
[F#]
[A#]
[A#]
[C#]
[A#]
[F#]

Figure 9. Notes selected to be joined as a chord.



☑ Onset ☑ Pitch ☑ Loudness ☑ Duration ☑ Voice

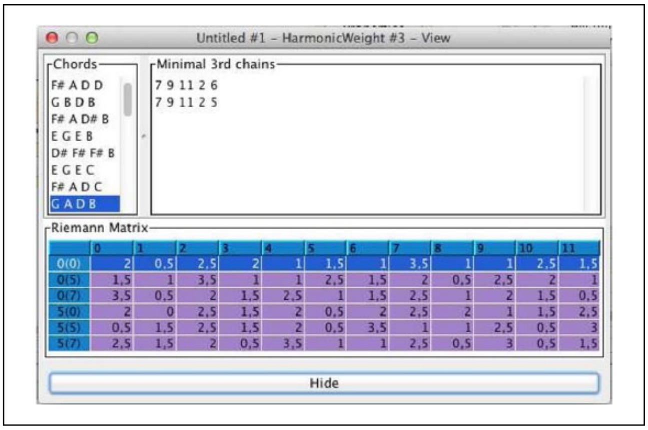
[C# F# A#]
[A#]
[C#]
[A#]
[F#]
[A#]
[C#]
[D#]

(0,00 ;C#5;82;1,88 ;0)
(0,00 ;F#5;80;1,88 ;0)
(0,00 ;A#5;80;1,88 ;0)

Figure 10. New chord sequence with duration and onset of notes updated by the user.

associating weights for a chord as a probability distribution in the three dimensional space of tonics (often identified with tonalities) (T), modes (M) and functions (F). Thus, the Riemann matrix table for a given chord Ch is a function $Weight(Ch): T \times M \times F \to \mathbb{R}_{\geqslant 0}$.

In the lower part of Figure 11, the Riemann matrix for the selected chord in the chord sequence can be seen. The calculation of a Riemann matrix is based on the *minimal third*



Chords

F# A D D
G B D B
F# A D# B
E G E B
D# F# F# B
E G E C
F# A D C
G A D B

Minimal 3rd chains

7 9 11 2 6
7 9 11 2 5

Riemann Matrix

	0	1	2	3	4	5	6	7	8	9	10	11
0(0)	2	0,5	2,5	2	1	1,5	1	3,5	1	1	2,5	1,5
0(5)	1,5	1	3,5	1	1	2,5	1,5	2	0,5	2,5	2	1
0(7)	3,5	0,5	2	1,5	2,5	1	1,5	2,5	1	2	1,5	0,5
5(0)	2	0	2,5	1,5	2	0,5	2	2,5	2	1	1,5	2,5
5(5)	0,5	1,5	2,5	1,5	2	0,5	3,5	1	1	2,5	0,5	3
5(7)	2,5	1,5	2	0,5	3,5	1	1	2,5	0,5	3	0,5	1,5

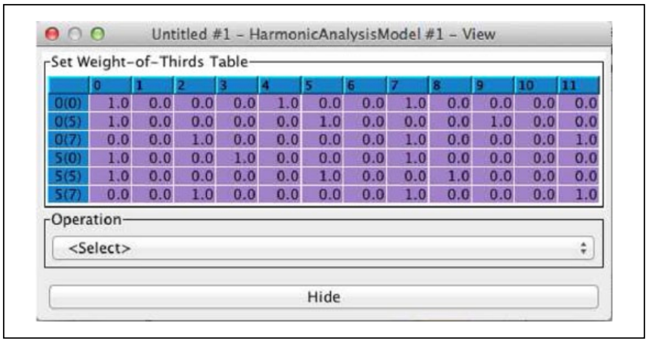
Figure 11. Minimal thirds chains and Riemann matrix values for the selected chord. The notes in the third chain remain within the span of an octave. Row names are encoded as *M(F)* where *M* is mode and *F* is function.

chain content of the chord *Ch*. This is defined as follows: a *third chain* is a chord *Th*, whose *n* elements can be ordered in a sequence Th_1, Th_2,...Th_n of pitch classes such that $Th_{i+1} = Th_i + s$, $i < n$, where $s = 3$ (minor third) or $s = 4$ (major third). A minimal third chain for *Ch* is a minimum length third chain, containing all pitches of the chord (each only once) and possibly containing other pitches not in the chord (see the Figure 11). Note that any possible chord can be embedded in at least one minimal third chain; in fact, every chord is part of the chromatic third chain {0, 3, 6, 9, 1, 4, 7, 10, 2, 5, 8, 11}. A list of all possible third chains ordered with respect to length is in the Topos of Music (Mazzola, 2002, pp. 1175–1181).

Let *Ch* be a chord. Let $Z = \{Z_1,...Z_n\}$ be the set of all minimal third chains Z_i for *Ch*. The Riemann matrix weights $Weight(Z_i)$ are calculated according to the values in the weight of thirds table in the rubette's properties. The values in the weight of thirds table can be edited by the user.

For our purposes we calculate an example. In Figure 11, for the selected chord {*G, A, D, B*}, all minimal third chains containing this chord are listed in the window as pitch class sequences. The first is 7–9–11–2–6, which when ordered to place neighboring notes a minor third or major third apart is 7–11–2–6–9. The second minimal thirds chain that contains the chord is 7–9–11–2–5.

By definition, the weight $Weight(Th)_{t,m,f}$ of a third chain *Th* for tonic *t* is weight of $Th - t$ (*Th* transposed down by *t*) for tonic *C*. This weight is the sum of the weights of its elements as defined in the table. For example, the average weight assigned to corresponding two chains (7–11–2–6–9 and 7–11–2–5–9) for *G* major tonic function is 3.5. For the first chain (7–11–2–6–9), in the thirds table shown in Figure 12, only {*G, A, D, F#*}, corresponding to pitch classes 7, 9, 2 and 6, are in G major with a weight of 1 each. For the second chain (7–11–2–5–9) only 7, 9 and 2 are present. The weight of the Riemann matrix in the first row and first column, which corresponds to G major tonality, is defined as the average of contributions for all minimal chains Z_i for the chord. In Figure 11 this average can be seen as 3.5: a contribution of 4 coming from the first chain and 3 from the second chain. Such a computation is done for every coefficient in the matrix.



Set Weight-of-Thirds Table

	0	1	2	3	4	5	6	7	8	9	10	11
0(0)	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
0(5)	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
0(7)	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
5(0)	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
5(5)	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0
5(7)	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0

Figure 12. Weight of thirds used to calculate Riemann matrices for chords. This table can be set through the HarmonicAnalysisModel rubette. This table is used for the Mozart analysis in the "Rubettes working together" section.

The weight of the chord *Ch* for a particular matrix point (*t, m, f*) is an average of all weights of minimal third chains for *Ch*:

$$weight(Ch)_{t,m,f} = \frac{1}{n}\sum_{i=1}^{n} weight(Z_i)_{t,m,f} \tag{1}$$

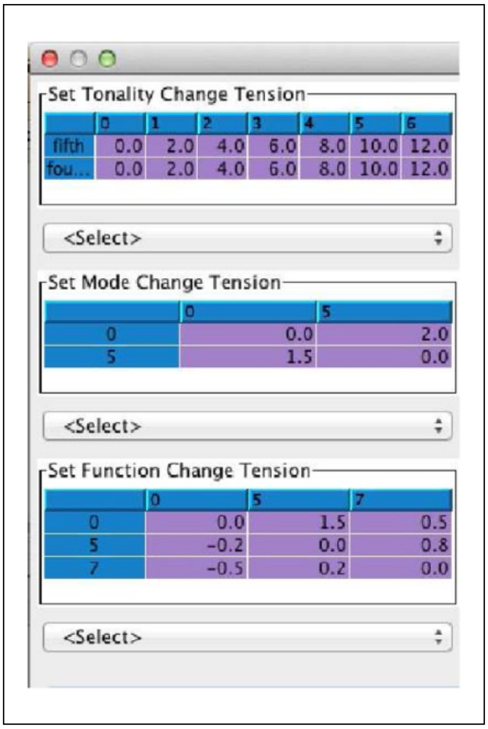
In the Riemann matrix high values correspond to a higher probability that the chord will be associated with that tonality, mode and function if other parameters are not taken into account. Riemann matrix values are evaluated in a context-free manner, regardless of the previous and successive chords and their associated functions in a tonality. The contextual consideration is done at the later stage when all possible harmonic paths are computed.

HarmonicPath rubette

Having calculated the Riemann matrix for each chord in a chord sequence, what is the tonality, mode and function to be attributed to every chord? As seen in Figure 11, every Riemann matrix has, at least, $card(T \times M \times F) = 72$ such "points". A *harmonic path* is a sequence $(p_1,\ldots,p_n)$ of points, one point p_i selected from each matrix M_i of the sequence $M_1,\ldots,M_n$ being calculated for the total chord sequence $Ch_1,\ldots,Ch_n$. For a given chord *Ch*, we shall also write $weight(p) = weight(Ch)_{t,m,f}$ where $p = (t, m, f)$ is a point of the Riemann matrix of *Ch*. Therefore, for *n* matrices, there are $(card(T \times M \times F))^n = 72^n$ paths to consider. If the search space is larger, e.g. all possible modes and functions are chosen via the HarmonicAnalysisModel rubette, then there are $(12 \times 7 \times 7)^n = 588^n$ paths. The *weight of a path* is defined as the sum of weights of its points and the distance between adjacent points (in adjacent matrices):

$$weight_{(p_1,\ldots,p_n)} = \sum_{i=1}^{n} weight(p_i) + \sum_{i=1}^{n-1} distance(p_i, p_{i+1}) \tag{2}$$

Each distance evaluation is the vector distance that uses the inverse values from tension tables in the rubette's property seen in Figure 13:


Set Tonality Change Tension

	0	1	2	3	4	5	6
fifth	0.0	2.0	4.0	6.0	8.0	10.0	12.0
fou...	0.0	2.0	4.0	6.0	8.0	10.0	12.0

<Select>

Set Mode Change Tension

	0	5
0	0.0	2.0
5	1.5	0.0

<Select>

Set Function Change Tension

	0	5	7
0	0.0	1.5	0.5
5	-0.2	0.0	0.8
7	-0.5	0.2	0.0

<Select>

Figure 13. Tension tables for updating tonality, mode, function tensions. This setup is used in the Mozart analysis in the "Rubettes working together" section.

$$distance(p_i, p_j) = e^{-[TonalityChange(ton_i,ton_j)+ModeChange(mod_i,mod_j)+FunctionChange(fun_i,fun_j)]} \tag{3}$$

The HarmonicPath rubette evaluates an optimal path, i.e. a path with maximum weight. Maximizing the weight in the formula (2) should correspond to maximizing weights of points and minimizing the tension between points. Minimizing distance in formula (3) corresponds to maximizing the inverse of the exponential of the vector distance.

In the HarmonicPath rubette's view dialog, the user can update tables for the tension created by moving from one matrix point to another point, that is to say, the tension created by changing tonality, mode and function. Therefore, finding the best (optimal) path means finding the path with maximal chord weight and minimal tension.

Tonality change tension is ordered according the cycle of fifths and fourths. For example, in Figure 13, tonalities that are a fifth or fourth apart have tonality tension of 2.0. When the distance in the cycle of fifths increases (clockwise or anticlockwise which corresponds to cycle of fourths) tension increases. The mode change tension table specifies the tension values when the mode changes: in this case when mode changes from major to minor and vice versa. For example, the tension created by going from major mode to minor mode is 2.0. Function change tension specifies the tension created between different functions. Tension created by going from tonic to subdominant is 1.0 (tension is increasing), whereas tension created by going from dominant to tonic is −0.5 (tension is decreasing).[3]

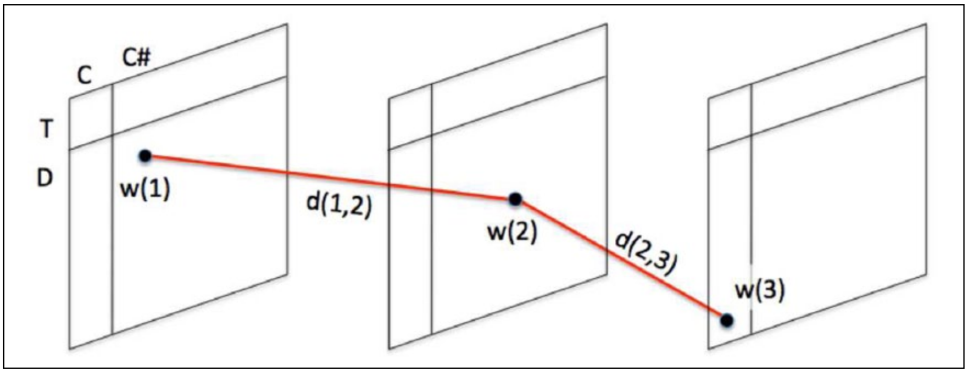


Figure 14. Three Riemann matrices in a window of size 3. A possible path is shown as a red line.

The practical problem in harmonic path calculation is the extreme size (72^n) of the search space (the number of possible paths) for *n* chords. To work around this problem a best path is evaluated locally, within a window. The window's size is given through *causal depth cd* and *final depth fd*, two quantities that together define the size of the context wherein a best path is evaluated. Thus, window size = $cd + fd + 1$.

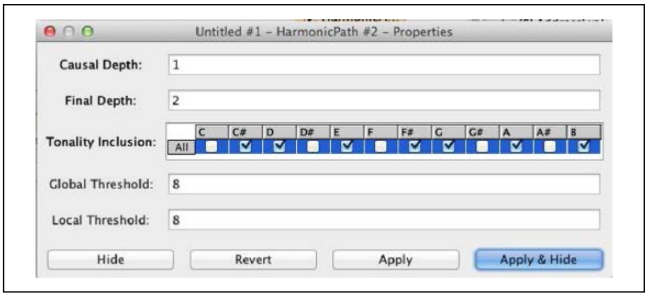
The algorithm for finding the path with maximum weight is as follows:

```
while there are more chords
  for the matrix m(c) (of chord c) in the current window
    find the path with maximum weight between matrices m-cd,m-cd+1,…,
    m,…,m+fd
    in the found path, the point p(m) of m is the
                            analysis result of chord c
  slide the window by one chord.
```

In order to automatically reduce the size of search space a "user configurable tonality exclusion table" is given, see Figure 15. Finally, *global* and *local thresholds* are percentage values, to be set for not taking into account matrix values with small weights relative to the local or global Riemann matrix maximum value. They are configured by the user in the rubette's properties. The output of the HarmonicPath rubette is the harmonic analysis result. The result is the sequence of chords, each chord being assigned a function in a tonality.

Harmonic path calculation in other reincarnations of HarmoRubette. Noll and Garbers also redesigned and reimplemented HarmoRubette (Noll & Garbers, 2004). They considered the problem of harmonic transitions among successive chords (dyadic transitions) as a first-order HMM (Noll & Garbers, 2004).

Our path model is not a HMM. The paths we define are only referring to the Riemann matrices, not the hidden chords. We also look to future values (the final depth), not only to the past as in the case of hidden Markov chains. Moreover, the optimization of path weights is not functional or even causal, it is rather a Hamiltonian principle as used in the Lagrangian formalism in physics. In Noll and Garbers' implementation, HarmoRubette remained a single component in the redesign, unlike our implementation where functionality is divided into four rubettes.



Untitled #1 – HarmonicPath #2 – Properties

Causal Depth: 1

Final Depth: 2

Tonality Inclusion:

All	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
	☐	✓	✓	☐	✓	☐	✓	✓	☐	✓	☐	✓

Global Threshold: 8

Local Threshold: 8

Figure 15. Setting window size, tonalities to exclude, and thresholds for Riemann matrix values. This setting is used for the Mozart analysis in the "Rubettes working together" section.

Noll and Garbers gave more control to the user to have access to all parts of the rubette, not only the tension settings for dyadic transitions as in the case of original HarmoRubette (Noll & Garbers, 2004). One way of access is during run-time through the scripting language, F-Script. Our network, doing the analysis in four consecutive steps, does not have such a run-time interface. In the original HarmoRubette, default settings for dyadic transitions are sensible for tonal music genre. In the redesign, more scientific control and responsibility in harmonic analysis is given to the user, both in our network and in Noll and Garbers' implementation. In their reimplementation, Noll and Garbers used the Viterbi algorithm for path finding (Noll & Garbers, 2004). The Viterbi algorithm is a practical algorithm to evaluate the most probable path through a set of states with associated probabilities and observed outcomes for each state (Forney, 1973). The algorithm's power is due to reducing computational complexity from N^T computations to N^2xT computations where N is the number of states for each step (in our case, it is the size of the Riemann Matrix) and T is the length of the sequence (in our case, it is the length of chord sequence). The Viterbi algorithm can be adopted for our path finding problem as well. In fact, when one thinks from a cognitive perspective, the way a human perceives chord functionality (considering also the temporal limitation due to auditory short-term memory) is similar to how the Viterbi algorithm works. That is to say, perception happens step by step in time, evaluating the cost of all incoming paths to the current state, and finding the optimal path as in the case of our algorithm. We find local maximal paths within the specified window and sequence them as the harmonic path. In our approach, the best path will be found only when the window size is the entire chord sequence. Thus, what our algorithm decides as the path may not correspond to the best path, which could be found using the Viterbi algorithm.

Rubettes working together

The following is a setup for harmonic analysis of a piece. The MidiFileIn rubette is an existing rubette in the Rubato Composer environment. Every rubette has a properties button and a view button. By clicking on the properties button the user provides parameters to control the running of the rubette. The view button is used to view the computed result, or to interact with the processed data.

The output of the MidiFileIn rubette outputs a denotator of score form (which is a denotator corresponding to a list of notes) which is the input to the ChordSequencer rubette. ChordSequencer's output is a sequence of chords and is the input to the HarmonicWeight rubette which evaluates a Riemann matrix for every chord in the sequence. The HarmonicWeight and the HarmonicPath rubettes both take as input the analysis specification from the HarmonicAnalysisModel rubette. This specification defines the Riemann matrix size and points. The input to the last rubette in the rubette network (see Figure 16), the HarmonicPath rubette, is the Riemann matrix sequence. The output of the HarmonicPath rubette is a table containing the result of the chord analysis. The rubettes' outputs can be displayed using the Display rubette, present in Rubato, as XML or in the denotator form.

An example: Mozart Piano Sonata in A

The process of harmonic analysis has been, in practice, a mixture of objective and subjective evaluation. Our aim is to make the analysis process well-defined and not only to make all decision parameters explicit, but also to bring them to the front. Such a parametrization and mechanization may help quantify the harmonic style. The fact that music-making uses intuition and aesthetic judgement does not mean that it can not be processed analytically.

From Mozart's Piano Sonata in A, K331/300i, first movement, the first 8 measures are shown in Figure 17. It is an 8-bar sentence in A major having a half-cadence in the middle and a perfect authentic cadence at the end. The Rubato analysis is as shown in Figure 18. The rubettes' analysis is performed with the properties in the ChordSequencer rubette and the HarmonicPath rubette and set as shown before; use-duration in the properties of the ChordSequencer rubette set to true. The total number of chords found through the ChordSequencer rubette is 36, based on the MIDI file that corresponds to the piece. In the HarmonicPath rubette, causal depth is one and final depth is two, and thus window size is four. This means that every chord's function is evaluated considering the previous chord and the upcoming two chords. The optimum path that these four chords define (among all possible paths) determines the chord's function. The values indicate function and mode in 'function(mode)' form. For example, the fourth chord in the analysis in Figure 18 is categorized as E:5(0) which is an encoding of form *Tonality:Function(Mode)*. E:5(0) means that the chord has a subdominant function in E major where 5 stands for subdominant function and 0 stands for major. The third chord, A:0(0) means the chord has tonic function in A major (0 is for tonic function, and 0 stands for major; 7 is for dominant function).

Discussion of analysis results. In Figure 18, the traditional analysis is notated on the score. The whole analysis is on a single key, A major with degrees I, II and V, without any modulation. In this traditional analysis, the first four bars (the first 19 chords) up to the half cadence, make the antecedent phrase. The next four bars is the consequent phrase, ending on a perfect authentic cadence.

In the network analysis in Figure 18, the 4th and 5th chords are labeled differently, although they have the same pitch content. The 4th chord, as seen on Figure 17, is classified as E:5(0), by the rubette analysis, i.e. it has a subdominant function in E major, i.e. A major. The reason that E:5(0) is chosen as opposed to A:0(0) is because in the window where E:5(0) is the second chord, E:5(0)–E:5(0)–E:5(0)–E:0(0) is the optimum path for the four-chord sequence.

Note that the algorithm decides about the chord in the window, not taking into account what was actually chosen before, i.e. although the chord before the present chord is A:0(0), during the decision, all possible previous tonalities and functions are considered. The 31st chord is analyzed as having a tonic function in E major. The path E:5(0)–E:0(0)–D:7(0)–D:0(0) is the optimal path in the window. That is why the result is E major. This aligns with the chord content: G#–E–B. The 32nd chord is analyzed as having a dominant function in D major. The path A:7(0)–D:7(0)–D:0(0)–D:7(0) is the optimal path in the window. Thus, the result is a

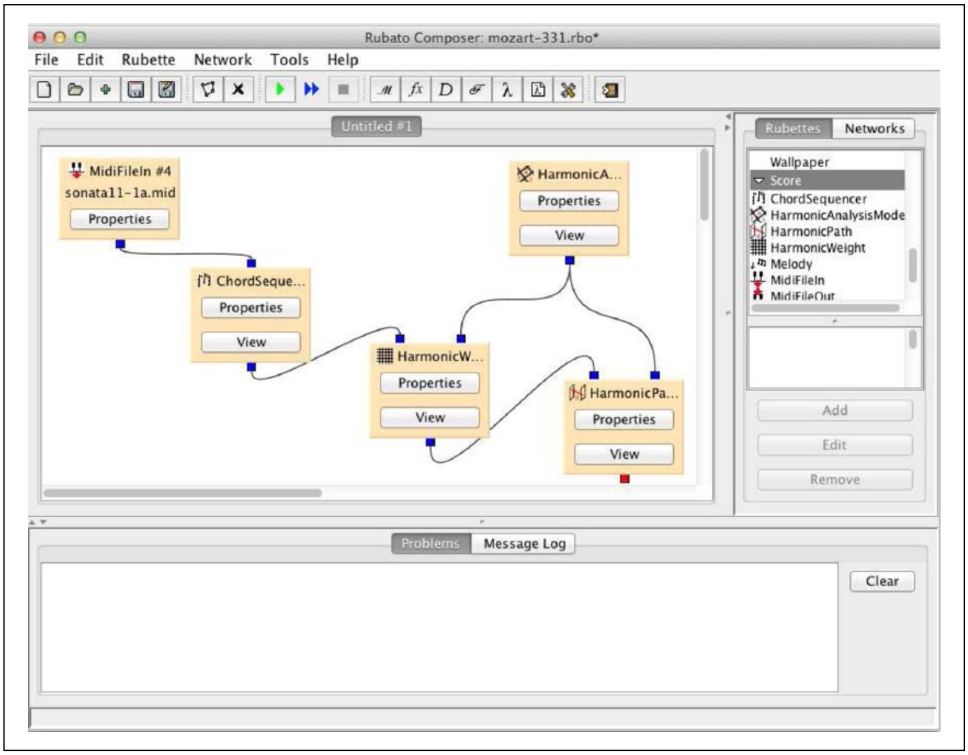


Figure 16. Five rubettes making a network for harmonic analysis.

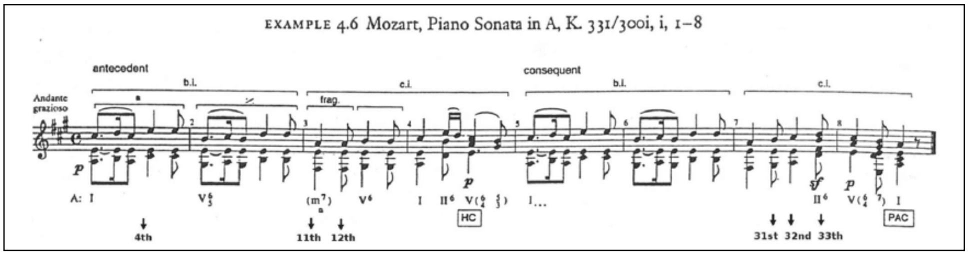
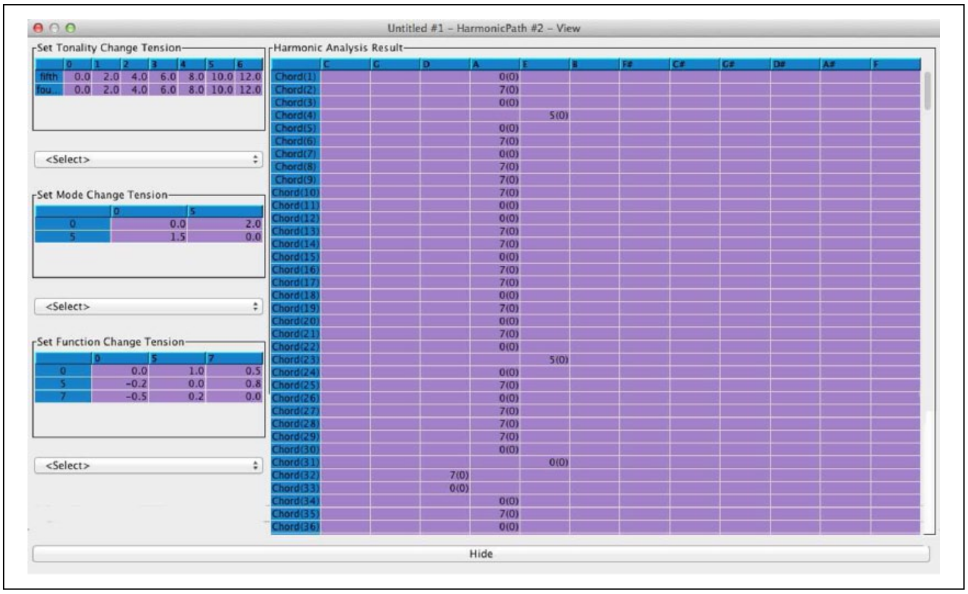


Figure 17. Mozart's Piano Sonata in A, K331/300i, i, 1–8. (Figure reproduced with permission from *Classical Form* by William E. Caplin, Oxford University Press, p. 52.)

dominant function in D major, as opposed to traditional analysis result which is tonic function in A major.

Another chord worth of attention is the 33rd chord which is classified as II of A major in the traditional analysis as seen in Figure 17, but D:0(0) by the rubette analysis, i.e. the tonic function in D major. In the case of this chord's analysis in the window, the optimal path is D:7(0)–D:0(0)–E:5(0)–E:0(0).

Now lets look at an example where traditional analysis and rubette analysis differs radically: consider the first two chords in the third bar, i.e. chords 11 and 12. The F#–E–A chord which is labeled as (m7), meaning an embellishing/non-structural minor 7th, in the traditional analysis



Set Tonality Change Tension

	0	1	2	3	4	5	6
fifth	0.0	2.0	4.0	6.0	8.0	10.0	12.0
fou...	0.0	2.0	4.0	6.0	8.0	10.0	12.0

Set Mode Change Tension

	0	5
0	0.0	2.0
5	1.5	0.0

Set Function Change Tension

	0	5	7
0	0.0	1.0	0.5
5	-0.2	0.0	0.8
7	-0.5	0.2	0.0

Harmonic Analysis Result

	C	G	D	A	E	B	F#	C#	G#	D#	A#	F
Chord(1)				0(0)								
Chord(2)				7(0)								
Chord(3)				0(0)								
Chord(4)					5(0)							
Chord(5)				0(0)								
Chord(6)				7(0)								
Chord(7)				0(0)								
Chord(8)				7(0)								
Chord(9)				7(0)								
Chord(10)				7(0)								
Chord(11)				0(0)								
Chord(12)				0(0)								
Chord(13)				7(0)								
Chord(14)				7(0)								
Chord(15)				0(0)								
Chord(16)				7(0)								
Chord(17)				7(0)								
Chord(18)				0(0)								
Chord(19)				7(0)								
Chord(20)				0(0)								
Chord(21)				7(0)								
Chord(22)				0(0)								
Chord(23)					5(0)							
Chord(24)				0(0)								
Chord(25)				7(0)								
Chord(26)				0(0)								
Chord(27)				7(0)								
Chord(28)				7(0)								
Chord(29)				7(0)								
Chord(30)				0(0)								
Chord(31)					0(0)							
Chord(32)			7(0)									
Chord(33)			0(0)									
Chord(34)				0(0)								
Chord(35)				7(0)								
Chord(36)				0(0)								

Figure 18. Rubette analysis of Mozart's Piano Sonata in A, K 331-330i, i, mm 1–8. Causal depth is one and final depth is two and thus window size is four.

in Figure 17. In fact, this result is difficult to find out from the chord's content due to missing C#. The motion in the bass over the whole bar F#–G#–A has significance. The rubette analysis computes the following path as optimum for the 11th chord: A:7(0)-A:0(0)-A:0(0)-E:0(0), i.e., Dominant of A major, Tonic of A major, Tonic of A major and Tonic of E major. The result is choosing an A major tonic function for the 11th chord. For the 12th chord, the result is the same however, A major tonic function. However, notice that a different optimum path is computed, i.e. A:0(0)–A:0(0)–A:7(0)–E:0(0), which is Tonic of A major, Tonic of A major, Dominant of A major and Tonic of E major. This is because, for an F#–E–A chord, A major gets higher weight than F#m7, and F# is a distant key in the region of A key.

It is not the goal of our approach to literally match traditional analytical annotations. First of all, the analysis of chord-slices for each onset constitutes a very basic level of description, which is not sensitive towards a distinction between proper harmonies and other verticalities, nor to the identification of broader harmonic units. But the study of the behavior of the automatically calculated analyses under systematic variation of the analytical parameters leads to a demanding experimental paradigm for empirical work. Eventually this may lead to alternative concepts of harmonic function on computational grounds.

Another example

The following example in Figure 19 is a music snippet where the K-S algorithm fails (Temperley, 2002). The passage is clearly not in E major, but due to the dominance of the pitch E, will be judged to be in key E by the K-S algorithm. Our network evaluates the passage as being in C major with the weight-of-thirds-table set as in Figure 20. Even if the tonic triad does not fall on the strong beat, it is still evaluated as C major. The reason that the evaluation does not change to E is due to the settings on the tonality change tension table. Pitch E, given the

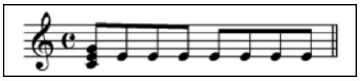


Figure 19. An example where the Krumhansl–Schmuckler key finding algorithm fails.

weight-of-thirds table as in Figure 20, contributes equally to C major and E major. When the weight-of-thirds table is changed to give more emphasis to the tonic (through the HarmonicAnalysisModel rubette), as in Figure 21, the evaluation changes to E major. (Note that the contribution of pitch E to E major and E minor are equal in the weight-of-thirds rubette. Thus, E minor and E major are equally weighted decisions by the analysis network.)

Comparison with other models

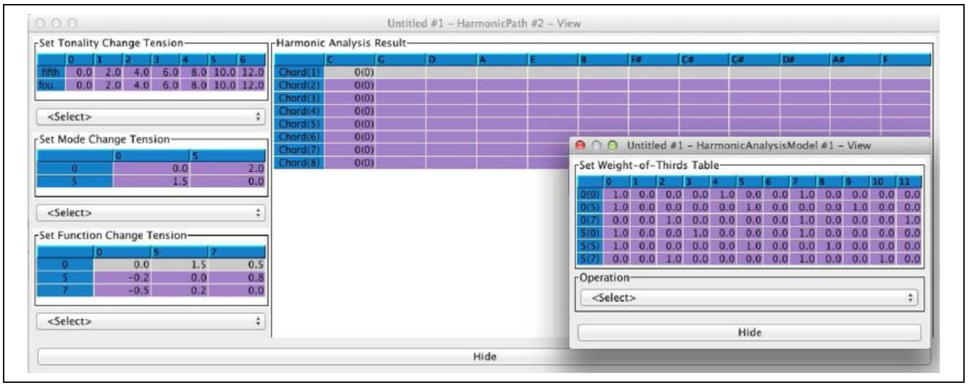
A comparison with Lerdahl's tension model

TPS is Fred Lerdahl's extension of GTTM. In TPS Lerdahl concentrates on pitch-related musical material and organization: chordal and regional spaces, harmonic prolongation, scales, chordal tension, melodic attraction. TPS differentiates between two different types of harmonic tension and defines a metric to calculate both: sequential (surface) tension and hierarchical (structural) tension. If x, y are triads, the distance $\delta(x, y)$ between x and y is the sum of differences in the triads based on the diatonic basic space and non-common pitch classes (Lerdahl, 2001). While surface tension is between adjacent chords, hierarchical tension is between distant chords, found only after the prolongational structure is revealed. Quantization of tension is used to calculate tonal tension for each chord in a tonal piece, taking into account both structural and sequential tension. The TPS tension model, along with the addition of melodic tension, is verified via experimental results based on listeners' emotional responses regulated via their expectations (Krumhansl & Lerdahl, 2010). The difficulty of using the TPS tension model for automated tonal analysis is the requirement that, to find the structural tension, one should already have made the structural analysis of the piece. Noll and Garbers also underline a theoretical problem with Lerdahl's chordal distance formula (Noll & Garbers, 2004).

Our tension model differs from Lerdahl's model in two ways. First our chords are not necessarily triads, but any set of notes. Second, our tension parameters are user-configurable, the default values being sensible with respect to tonal music. For example, all other things being equal, moving from tonic to dominant increases tension, and the reversed movement decreases tension. Moving from one diatonic basic space to another also increases tension directly proportional to the cycle-of-fifths distance of the spaces.

A comparison with Sapp's "stack of thirds" approach

Sapp's computational chord-root identification, which he considers as the first step towards AHA, arranges chords as a "stack of thirds". To find the root of the chord, he considers all possible such stacks, and gives a compactness score for every combination (Sapp, 2007). Sapp's consideration of minor 3rd and major 3rd intervals as primary intervals for tonal music of the common practice period also supports our minimal-3rd-chain approach.


Untitled #1 – HarmonicPath #2 – View

Set Tonality Change Tension

	0	1	2	3	4	5	6
fifth	0.0	2.0	4.0	6.0	8.0	10.0	12.0
fou...	0.0	2.0	4.0	6.0	8.0	10.0	12.0

<Select>

Set Mode Change Tension

	0	5
0	0.0	2.0
5	1.5	0.0

<Select>

Set Function Change Tension

	0	5	7
0	0.0	1.5	0.5
5	-0.2	0.0	0.8
7	-0.5	0.2	0.0

<Select>

Harmonic Analysis Result

	C	G	D	A	E	B	F#	C#	G#	D#	A#	F
Chord(1)	0(0)											
Chord(2)	0(0)											
Chord(3)	0(0)											
Chord(4)	0(0)											
Chord(5)	0(0)											
Chord(6)	0(0)											
Chord(7)	0(0)											
Chord(8)	0(0)											

Hide

Untitled #1 – HarmonicAnalysisModel #1 – View

Set Weight-of-Thirds Table

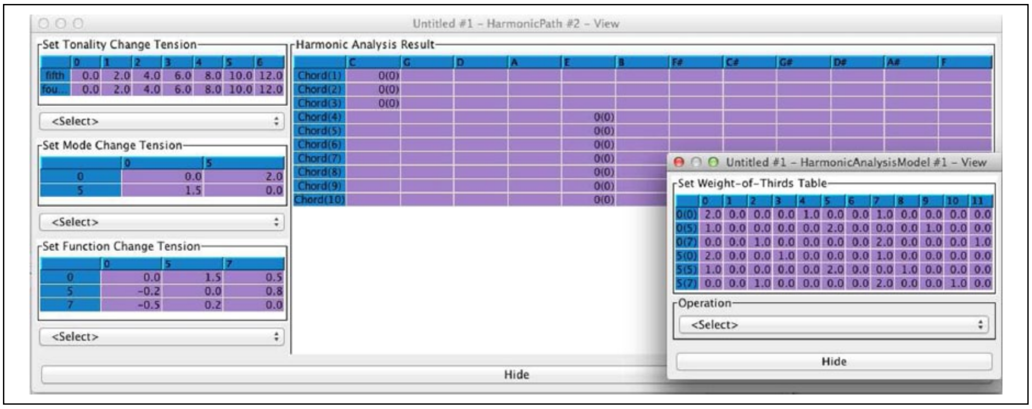
	0	1	2	3	4	5	6	7	8	9	10	11
0(0)	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
0(5)	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
0(7)	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
5(0)	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
5(5)	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0
5(7)	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0

Operation

<Select>

Hide

Figure 20. The weight-of-thirds table used and the analysis results of the Harmonic Analysis Network for analysis of the music snippet in Figure 19.


Untitled #1 – HarmonicPath #2 – View

Set Tonality Change Tension

	0	1	2	3	4	5	6
fifth	0.0	2.0	4.0	6.0	8.0	10.0	12.0
fou...	0.0	2.0	4.0	6.0	8.0	10.0	12.0

<Select>

Set Mode Change Tension

	0	5
0	0.0	2.0
5	1.5	0.0

<Select>

Set Function Change Tension

	0	5	7
0	0.0	1.5	0.5
5	-0.2	0.0	0.8
7	-0.5	0.2	0.0

<Select>

Harmonic Analysis Result

	C	G	D	A	E	B	F#	C#	G#	D#	A#	F
Chord(1)	0(0)											
Chord(2)	0(0)											
Chord(3)	0(0)											
Chord(4)					0(0)							
Chord(5)					0(0)							
Chord(6)					0(0)							
Chord(7)					0(0)							
Chord(8)					0(0)							
Chord(9)					0(0)							
Chord(10)					0(0)							

Hide

Untitled #1 – HarmonicAnalysisModel #1 – View

Set Weight-of-Thirds Table

	0	1	2	3	4	5	6	7	8	9	10	11
0(0)	2.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
0(5)	1.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	1.0	0.0	0.0
0(7)	0.0	0.0	1.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	1.0
5(0)	2.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
5(5)	1.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	1.0	0.0	0.0	0.0
5(7)	0.0	0.0	1.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	1.0	0.0

Operation

<Select>

Hide

Figure 21. The weight-of-thirds table used and the analysis results of the Harmonic Analysis Network for analysis of the music snippet in Figure 19. The weight-of-thirds table where tonic's weight is doubled in triads, getting closer to Krumhansl key-profile weights.

Future directions

A second pass for hierarchy

What our network does for harmonic analysis is enough to identify chords as tokens, and to have an overall sense of the key. Note that it finds the perfect authentic cadence in Mozart successfully as well. Turning chords into tokens is exactly like turning character sequences to tokens (lexemes) in a compiler's first phase, i.e. the scanner (lexer) phase. Taking the analogy further, in the second phase, the parser builds a parse tree from the sequence of tokens.

There are two things to consider in the second pass.

- We do not take into account rhythm. A chord that has a strong accent, or falls on a strong beat, is not given more importance as compared to a chord that falls on a weak beat.

- The window size used in path calculation is not dynamic, but fixed by the user prior to analysis. A dynamic windows size will be more efficient in capturing prolonged harmonies.

A second usage of harmonic analysis is to find out how important a note or a group of notes is for the overall harmony. Omitting a note and finding out how the overall harmonic analysis change would be a measure of the harmonic importance of the note. Knowing the importance of note(s) in a chord and in the overall harmony, the performer may decide to emphasize these notes. This has been implemented for the harmonic weight calculation of single notes in the original HarmoRubette. A third enhancement would be to make the weight formula for paths, i.e. equation (2), user-configurable. The user could then choose among different weight formulas and also be able to add their own weight formula. This is implemented by Noll and Garbers (2004). A fourth enhancement could allow different weight calculation for chords, not necessarily limited to minor and major third chains. In this way, new harmonic approaches could be analyzed such as post-tonal music with set theory-based analytical tools, or quarter-tone music, or non-Western harmonic theories. Although not currently implemented, in principle our HarmonicAnalysisModel rubette allows not only a 12-tone equal-tempered scale, but can work on any scale, with adequate metrics for chord calculation and chord transitions.

Conclusion

Parameterization of harmonic analysis shows that there is not one *ideal* analysis of a piece. Any "ideal" or "good" analysis is one with specific preconceptions, with a particular set of parameters. Traditionally, one such parameter has been the minimum number of key changes which corresponds to having tonality change tension high and function change tensions low. We thought of calling the set of parameters that control the analysis a *harmonic perspective* since different perspectives result in different views, i.e. different analysis results.

The Harmonic Analysis Network can be used for the analysis of harmony in tonal music. We tried to have the least amount of a priori assumptions and provide almost full control of the analysis procedure to the user, if the user wishes so. However, default values for tension and weight-of-thirds table provide sensible analysis criteria for common-practice tonal music.

Acknowledgements

The authors would like to thank the anonymous referees for their comments and suggestions which improved this work. The authors would also like to thank Dr James Holdman who proofread this paper thoroughly and improved the written presentation enormously.

Funding

One of the authors (R.A.) is partially supported by Istanbul Bilgi University.

Notes

1. Available at: http://rubato.org/rubettes.html
2. Download from http://rubato.org/rubettes.html
3. We use imaginary number representation to be able to calculate vector distance for negative tension. Thus, −0.2 in the table corresponds to $0.2i$ in our computation.

References

Bas De Haas,W., Magalhães, J.P., Wiering, F. & Veltkamp, R.C. (2011). *Automatic functional harmonic analysis* (Technical Report UU-CS-2011-023). Utrecht, The Netherlands: Department of Information and Computing Sciences Utrecht University.

Bigand, E., Delbé, C., Poulin-Charronnat, B., Leman, M. & Tillmann, B. (2014). Empirical evidence for musical syntax processing? Computer simulations reveal the contribution of auditory short-term memory. *Frontiers in Systems Neuroscience, 8*(4), 94.

Caplin, W. (Ed.). (1998). *Classical form* (1st ed.). New York: Oxford University Press.

Chew, E. (2001). Modeling tonality: Applications to music cognition. In J. D. M. K. Stenning (Ed.), *Proceedings of the 23rd annual meeting of the cognitive science society* (pp. 206–211). Mahwah, NJ: Lawrence Erlbaum Assoc. Pub.

Christensen, T. (Ed.). (2006). *The Cambridge history of Western music theory* (1st ed.). New York: Cambridge University Press.

Dahlhaus, C. (1966). *Über den Begriff der tonalen Funktion.* (M. Vogel, Ed.). Regensburg, Germany: Gustav Bosse.

Forney, G.D., Jr. (1973). The viterbi algorithm. In *Proceedings of the IEEE* (Vol. 61, pp. 268–278). New York: IEEE.

Hamanaka, M., Hirata, K., & Tojo, S. (2005). *ATTA: Automatic time-span tree analyzer based on extended GTTM.* Retrieved from http://ismir2005.ismir.net/proceedings/index.html

Hyer, B. (1995). Re-imagining Riemann. *Journal of Music Theory, 39*(1), 101–138.

Krumhansl, C. (1990). *Cognitive foundations of musical pitch.* New York: Oxford University Press.

Krumhansl, C., & Lerdahl, F. (2010). Musical tension. In F. Bacci & D. Melcher (Eds.), (pp. 297–313). New York: Oxford University Press.

Lerdahl, F. (2001). *Tonal pitch space.* New York: Oxford University Press.

Lerdahl, F., & Jackendorff, R. (1983). *A generative theory of tonal music.* Cambridge, MA: MIT Press.

Lerdahl, F., & Krumhansl, C. (2007). Modeling tonal tension. *Music Perception, 24*(4), 329–366.

Mazzola, G. (2002). *The topos of music.* Basel, Switzerland: Birkhäuser.

Milmeister, G. (2009). *The Rubato Composer music software.* Berlin, Germany: Springer-Verlag.

Noll, T., & Garbers, J. (2004). Harmonic path analysis. In T. N. G. Mazzola & E. Luis-Puebla (Eds.), *Perspectives of mathematical and computational music theory.* Osnabrück, Germany: epOs-Music.

Patel, A. (2008). *Music, language and the brain.* New York: Oxford University Press.

Riemann, H. (1992). Ideas for a study "on the imagination of tone". *Journal of Music Theory, 36*, 81–117.

Sapp, C. (2007). Computational chord-root identification in symbolic musical data: Rationale, methods and applications. *Computing in Musicology, 15*, 99–119.

Sleator, D., & Temperley, D. (2003). *The Melisma Music Analyzer.* Retrieved from http://www.link.cs.cmu.edu/music-analysis/

Taube, H. (1991). Automatic tonal analysis: Toward the implementation of a music theory workbench. *Computer Music Journal, 23*(4), 18–32.

Taube, H., & Burnson, W. (2009). *Software for teaching music theory.* Published online by Michigan Publishing of University of Michigan Library. Retrieved from http://quod.lib.umich.edu/i/icmc/bbp2372.2009?rgn=full+text

Temperley, D. (2002). A Bayesian approach to key-finding. In C. Anagnostopoulou, M. Ferrand & A. Smaill (Eds.), *ICMAI 2002, LNAI 2445* (pp. 195–206). Berlin, Germany Springer-Verlag.

Winograd, T. (1968). Linguistics and the computer analysis of tonal harmony. *Journal of Music Theory, 12*(1), 2–49.