

Chapter 2

The Harmonic Musical Surface and Two Novel Chord Representation Schemes

Emilios Cambouropoulos

Abstract Selecting an appropriate representation for chords is important for encoding pertinent harmonic aspects of the musical surface, and, at the same time, is crucial for building effective computational models for music analysis. This chapter, initially, addresses musicological, perceptual and computational aspects of the harmonic musical surface. Then, two novel general chord representations are presented: the first, the General Chord Type (GCT) representation, is inspired by the standard Roman numeral chord type labelling, but is more general and flexible so as to be applicable to any idiom; the second, the Directed Interval Class (DIC) vector, captures the intervallic content of a transition between two chords in a transposition-invariant idiom-independent manner. Musical examples and preliminary evaluations of both encoding schemes are given, illustrating their potential to form a basis for harmonic processing in the domain of computational musicology.

2.1 Introduction

Research in computational musicology and, more specifically, computational music analysis commonly assumes the fundamental concept of the *musical surface*, i.e., a minimal discrete representation of the musical sound continuum in terms of note-like events (each note described by pitch, onset, duration, and possibly dynamic markings and timbre/instrumentation). The musical surface is assumed to be merely an unstructured sequence of atomic note events, such as score notes or a piano-roll representation. Taking as a starting point this elementary musical surface, abstract structures may be determined, such as grouping/segmentation, metre, chords and motivic categories.

Emilios Cambouropoulos
School of Music Studies, Aristotle University of Thessaloniki, Thessaloniki, Greece
e-mail: emilios@mus.auth.gr

In this chapter we will focus on aspects of the musical surface that pertain to musical harmony. Challenging the ‘standard’ understanding (at least in the domain of computational musicology) of the musical surface as being the note level of a musical piece, it will be maintained that chords as wholes should be considered as an integral part of the musical surface. The emergence of this harmonic musical surface involves rather complex mechanisms that require, among other things, consonance/dissonance discrimination, chord-type abstraction, root-finding and function categorization (leaving aside non-harmonic factors such as rhythm, melody and voice separation).

A novel general representation of chord types is proposed that is appropriate for encoding tone simultaneities in any harmonic context whether it be tonal, modal, jazz, octatonic or even atonal. This *General Chord Type* (GCT) representation allows for the rearrangement of the notes of a harmonic simultaneity or pitch set, such that abstract, idiom-specific types of chords may be derived. The GCT algorithm finds the maximal subset of notes of a given note set that contains only consonant intervals, employing a user-specified consonance/dissonance classification; this maximal subset forms the base upon which the chord type is built. The proposed representation is ideal for hierarchic harmonic systems such as the tonal system and its many variations (actually the GCT is designed such that properties of the standard Roman-numeral encoding scheme are naturally accommodated), but adjusts to any other harmonic system such as post-tonal, atonal music, or traditional polyphonic systems. It thus allows for automatic chord-type labelling (resembling traditional Roman numeral encoding) in diverse musical idioms. The application of the GCT algorithm is illustrated on a small set of examples from a variety of idioms and tested on the Kostka–Payne harmonic dataset (Temperley, 2001b).

A proposal for representing chord transitions in an idiom-independent manner is also introduced. A harmonic transition between two chords can be represented by a *Directed Interval Class* (DIC) vector (this is an adaptation of Lewin’s *interval function* between two collections of notes—see Sect. 2.4). This representation allows for the encoding of chord transitions at a level higher than individual notes that is transposition-invariant and idiom-independent (analogous to pitch intervals that represent transitions between notes). The proposed 12-dimensional vector encodes the frequency of occurrence of each distinct *directional* interval class (from 0 to 6 with +/– for direction) between a pair of notes in two successive chords. Apart from octave equivalence and interval inversion equivalence, this representation preserves directionality of intervals (up or down). The proposed DIC representation has been evaluated on a harmonic recognition task (specifically, the identification of harmonic queries in a small database consisting of pieces from diverse idioms). The DIC vector representation is very general and may be useful in tasks such as chord pattern recognition tasks, but is rather too abstract to be used in tasks such as sophisticated harmonic analysis or melodic harmonization (for which the GCT representation is more appropriate).

In the following sections, the notion of musical surface will first be discussed in perceptual, musicological and computational terms. Then the GCT representation will be described. Finally, the DIC vector chord transition encoding will be presented.

2.2 The Harmonic Music Surface and Chord Representation Schemes

In this section, it is maintained that note simultaneities are perceived as chord types (e.g., major, minor, dominant seventh, etc.) prior to establishing more elementary aspects such as individual pitch octave information, note doubling, note omission and chord inversion. This intuition is reflected in established musicological theoretical typologies such as the standard guitar-like chord encoding or Roman numeral analytic labels or even pc-set categories. Advantages and shortcomings of such chord formalisms will be discussed primarily in relation to computational music analysis. A more extended discussion on representing the musical surface is given by Cambouropoulos (2010).

Jackendoff (1987), by analogy with the linguistic surface of phonemes, defines the *musical surface* as being the “lowest level of representation that has musical significance” (p. 219) and suggests that “standard musical notation represents the pitch-events of the musical surface by means of symbols for discrete pitch and duration” (p. 218). Sloboda (1985) suggests that the “basic ‘phoneme’ of music is a ‘note’” (p. 24) and presents empirical evidence for categorical perception of pitch and duration. Should the *note*, however, be considered as the lowest level of representation that has musical significance and perceptual relevance?

In terms of co-sounding events, there is evidence that pitch intervals and chords are commonly perceived by listeners in a holistic manner, prior to their being perceived in terms of their constituent parts. Over eighty years ago, Vernon (1934), a Gestalt psychologist, suggested that ordinary listeners frequently perceive holistically a more or less complex auditory figure, such as a complex tone or a chord, without knowing or being able to analyse its constituent elements. Empirical research has shown that listeners perceive musical intervals categorically (see Burns, 1999; Handel, 1989; Smith et al., 1994). Categorical perception applies to chords as well, as has been shown by Locke and Kellar (1973). It is suggested that pitch intervals or chords are actually closer to the categorically perceived phonemic units of language than isolated notes.

Due to octave equivalence and transpositional pitch interval equivalence, harmonic pitch intervals and chords are considered ‘equivalent’ even though their constituent pitches may be placed in different octaves. Empirical research has shown that listeners confuse pairs of tones that are related by inversion (Deutsch, 2012) and that chord positioning does not affect recognizing components of a chord—i.e., that chords in different positions are essentially equivalent (Hubbard and Datter, 2001).

Parncutt (1989) provides a psychoacoustic explanation of how chords are heard. Just as our perceptual mechanisms analyse a complex periodic sound into partials and then re-integrate them into a single percept, so chords can be heard as a single entity with a single perceived root, rather than three or more individual co-sounding tones. Parncutt suggests that the same factors that govern the perception of individual pitches govern the perception of chords. Parncutt (1997) proposes an extended model that calculates the perceptual root of a chord from its pitch classes, voicing, and

the prevailing tonality. Parncutt's approach to chord perception is in line with the suggestion in this chapter that chords are perceived at the surface level as single integrated entities rather than sets of constituent atomic notes.

Identifying a set of co-sounding partials or notes as, for instance, a major, minor or diminished chord involves additional culture-specific knowledge that is acquired via exposure to a certain idiom. "Chord recognition is the result of a successful memory search in which a tone series is recognized as a pattern stored in long-term memory" (Povel and Jansen, 2001, p. 185). Template-matching models (Parncutt, 1994) are integrated in cognitive mechanisms of musical listening and are responsible for the extraction of musically pertinent entities from sound at the surface level (the musical surface is specific to a musical idiom in a similar way to that in which phonological structure is language-specific). The above discussion supports the idea that chords tend to be perceived at the surface level as single integrated entities, rather than agglomerates of independent atomic notes.

The reduction of the sound continuum into more abstract (symbolic) entities such as notes, chords, trills, and so on, may be attributed to general human cognitive mechanisms that aim to reduce sensory information into smaller more manageable discrete quantities (categorical perception). What are the grounding principles that enable this reduction? More specifically, what are the principles that allow the integration/segregation of distinct harmonics/tones into single entities or coherent 'wholes'? The basic perceptual mechanisms that enable the breaking down of the acoustic signal into more manageable units and successions of units (streams) have been investigated extensively in the field of auditory scene analysis (Bregman, 1994). These principles can be applied or adapted to account for musical practices of voice separation/integration and voice leading (Huron, 2001). For instance, principles such as tonal fusion, onset synchrony and pitch co-modulation (Huron, 2001) may play an important role in the fusion of co-sounding entities into larger percepts such as chords (e.g., greatest fusion in parallel motion of octave-related tones).

In recent years, a number of voice separation algorithms have emerged; these algorithms mostly attempt to separate polyphonic unstructured note complexes into a number of monophonic voices (see, e.g., Jordanous, 2008, and Chap. 6, this volume). Cambouropoulos (2008) adopts a different approach in which multi-note sonorities are allowed within individual 'voices'. Allowing both horizontal and vertical integration allows the algorithm to perform well not only in polyphonic music that has a fixed number of 'monophonic' lines, but in the general case where both polyphonic and homophonic elements are mixed together. In this sense, the musical surface is considered as consisting of both simple notes organized in melodic streams, and chords organized in chordal streams (such streams may appear independently, or in parallel, or may overlap).

A common underlying assumption in much cognitive and computational modelling of musical understanding is that musical structural processing starts at the musical surface and proceeds towards higher structural levels, such as metre, rhythmic patterns, melodic motives, harmonic structure and so on. Lerdahl and Jackendoff's (1983) influential theory is grounded on this assumption:

The musical surface, basically a sequence of notes, is only the first stage of musical cognition. Beyond the musical surface, structure is built out of the confluence of two independent hierarchical dimensions of organization: rhythm and pitch.

(Jackendoff and Lerdahl, 2006, p. 37)

It is often an underlying assumption in computational research that from audio the score may be extracted and then higher-level processing is possible.

However, as Cemgil et al. (2006) have pointed out,

one of the hard problems in musical scene analysis is automatic music transcription, that is, the extraction of a human readable and interpretable description from a recording of a music performance.

Indeed, research in automated music transcription has shown that a purely bottom-up approach (from audio to score) is not possible; higher-level music processing is necessary to assist basic multi-pitch and onset extraction techniques so as to reach acceptable transcription results. Ryyänen and Klapuri (2008, p. 73) have observed that

nowadays the concept of automatic music transcription includes several topics such as multipitch analysis, beat tracking and rhythm analysis, transcription of percussive instruments, instrument recognition, harmonic analysis and chord transcription, and music structure analysis.

Some of the ‘higher level’ musical processes that are necessary for transcription are addressed by Cambouropoulos (2010). It is suggested, not only that higher-level processing influences the formation of the musical surface, but that some processes that are considered ‘higher-level’ are actually necessary for the formation of the surface per se, which means, essentially, that they are at or below the musical surface. It is maintained that, for instance, beat structure, chord simultaneities and voice separation are internal ‘primitive’ processes of the musical surface, that are necessary for the surface to emerge.

For the sake of simplicity, in this chapter, we deal solely with purely symbolic, isorhythmic, homophonic textures (i.e., sequences of chords without secondary embellishment notes). Harmonic reduction (i.e., the abstraction of main chord notes from a musical work) is anything but a trivial task. Reduction relies not only on rhythm, metric position and melodic qualities (e.g., passing or neighbour notes), but also on harmony per se. That is, harmonic knowledge is paramount in establishing which notes are secondary and can be omitted. For instance, access to previously learned harmonic context in a given idiom (chord patterns) may assist the selection of appropriate chords that give rise to acceptable chord progressions (Mauch et al., 2010). The representation schemes presented in the following sections (especially the GCT) can be extended in the future so as to facilitate the recognition of harmonies in an unreduced collection of notes or unreduced stream of sounds (this is beyond the scope of the current chapter).

Researchers that work on symbolic music data (e.g., quantized MIDI or encodings of scores) commonly assume that the formation of the musical surface (i.e., the notes) from audio requires a potentially large amount of processing; once, however,

the surface is formed, the road to higher-level processing (such as beat tracking, metre induction, chord analysis, pattern extraction, and so on) is open. On the other hand, researchers that work on music audio often leave aside the whole question of musical surface and attempt to extract high-level information (e.g., harmonic patterns, structural segmentation, music similarity, cover song identification) directly from audio. We suggest that the whole discussion on the musical surface, apart from being of theoretical interest, may make researchers more aware of the need to think more carefully when deciding which primitive starting representation to use for their systems (audio, expressive or quantized MIDI, pitch classes, notes, chords, etc.); how much processing is already implicitly ‘embodied’ in this primitive representation; and what kind of information can be extracted ‘naturally’ from the selected starting representation.

Computational systems developed for harmonic analysis and/or harmonic generation (e.g., melodic harmonization), rely on chord labelling that is relevant and characteristic of particular idioms. There exist different typologies for encoding note simultaneities that embody different levels of harmonic information/abstraction and cover different harmonic idioms. For instance, for tonal musics, chord notations such as the following are commonly used: figured bass (pitch classes denoted above a bass note—no concept of ‘chord’); popular music guitar-style notation or jazz notation (absolute chord); and Roman numeral encoding (chord function relative to a key) (Laitz, 2012). For atonal and other non-tonal systems, pc-set theoretic encodings (Forte, 1973) may be employed.

For computational models of tonal music, Harte et al.’s (2005) representation provides a systematic, context-independent syntax for representing chord symbols which can easily be written and understood by musicians, and, at the same time, is simple and unambiguous to parse with computer programs. This chord representation is very useful for manually annotating tonal music—mostly genres such as pop, rock and jazz that use guitar-style notation. However, it cannot be automatically extracted from chord reductions and is not designed to be used in non-tonal musics.

Two questions are raised and addressed in the remainder of this chapter. First, is it possible to devise a ‘universal’ chord representation that captures features of hierarchic pitch systems and adapts to different harmonic idioms? Is it possible to determine a mechanism that, given some fundamental idiom features, such as pitch hierarchy and consonance/dissonance classification, can automatically abstract chord types and encode pitch simultaneities in a pertinent manner for the idiom at hand? A second question regards chord transitions: is a relative pitch encoding of chords possible such that chord *transitions* (i.e., intervallic content) are captured without recourse to constituent chords? The first question will be addressed in the next section, the second in Sect. 2.4.

It should be noted that the representations and processes reported in the following sections do not explicitly deal with the issue of harmonic surface, but rather with two relatively simple schemes of chord encoding that capture different aspects of the harmonic surface. No explicit cognitive claims are made; however, it is suggested that these representations may capture some properties of chords that are cognitively relevant and are potentially linked to the notion of the harmonic surface. The main

objective of the proposed representations is to provide appropriate, general encodings of aspects of the harmonic surface that may be useful in various computational music-analytic tasks, such as harmonic analysis, chord sequence similarity, and harmonic recognition. Hopefully, the proposed representations may be incorporated in more sophisticated music surface extraction and processing schemes, providing a useful step that links the sub-symbolic domain with higher-level musical structure.

2.3 The General Chord Type (GCT) Representation

Harmonic analysis focuses on describing the harmonic content of pitch collections and patterns within a given music context in terms of harmonic labels, classes, functions and so on. Harmonic analysis is a rather complex musical task that involves not only finding roots and labelling chords within a key, but also segmentation (points of chord change), identification of non-chord notes, metric information and other aspects of musical context (Temperley, 2001a, 2012). In this section, we focus on the core problem of labelling chords within a given pitch hierarchy (e.g., key). We assume, for simplicity, that a full harmonic reduction, identifying the main harmonic notes, is available as input to the model along with key and modulation annotations.

In trying to tackle issues of tonal hierarchy, a novel representation of chord types is proposed that is appropriate for encoding tone simultaneities (or more generally pitch sets) in any harmonic context (whether it be tonal, modal, jazz, octatonic or even atonal). The *General Chord Type* (GCT) representation, allows the rearrangement of the notes of a harmonic simultaneity such that abstract, idiom-specific types of chords may be derived; this encoding is inspired by the standard Roman numeral chord type labelling, but is more general and flexible. Given a consonance/dissonance classification of intervals (that reflects sensory and/or culturally-dependent notions of consonance and dissonance), and a scale, the GCT algorithm finds the maximal subset of notes of a given note simultaneity that contains only consonant intervals; this maximal subset forms the base upon which the chord type is built. The lowest note of the base is the root of the chord. The proposed chord type representation, takes as its starting point the common-practice tonal chord representation (for a tonal context, it is equivalent to the standard Roman-numeral harmonic encoding), but is more general as it can be applied to other non-standard tonal systems such as modal and even atonal harmony. This representation is based on notions drawn primarily from the domain of psychoacoustics and music cognition (such as octave equivalence, root, relative root, consonance, parsimony), and, at the same time, ‘adjusts’ to different culture-specific contexts of scales, tonal hierarchies and rankings of consonance and dissonance. Cambouropoulos et al. (2014) provide a more extended discussion on the background concepts underlying the GCT model.

At the heart of the GCT representation is the idea that the *base* of a note simultaneity should be ‘consonant’ (within some particular musical idiom). The GCT algorithm tries to find a maximal subset that is consonant; the rest of the notes that create dissonant intervals to one or more notes of the chord base form the chord

extension. The GCT representation has common characteristics with the stack-of-thirds and the virtual pitch root-finding methods for tonal music, but has differences as well (see Cambouropoulos et al., 2014). Moreover, the user can define which intervals are considered consonant; thus giving rise to different encodings. As will be shown in the following subsections, the GCT representation naturally encapsulates the structure of tonal chords and, at the same time, is very flexible and can readily be adapted to different harmonic systems.

2.3.1 Description of the GCT Algorithm

Given a classification of intervals into the categories consonant and dissonant (i.e., binary values) and an appropriate scale background (i.e., scale with tonic), the GCT algorithm computes, for a given multi-tone simultaneity, the ‘optimal’ ordering of pitches such that a maximal subset of consonant intervals appears at the base of the ordering (left-hand side) in the most compact form. Since a tonal centre (key) is given, the position within the given scale is automatically calculated.

The input to the algorithm is the following:

- | | |
|------------------------------|--|
| <i>Consonance vector</i> | The user defines which intervals are consonant/dissonant. A 12-dimensional vector is employed in which each value corresponds to a pitch interval from 0 to 11 (in the current version of the algorithm, Boolean values are used (i.e., consonant=1, dissonant=0)). ¹ For instance, the vector [1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0] means that the unison, minor and major third, perfect fourth and fifth, minor and major sixth intervals are consonant; dissonant intervals are the seconds, sevenths and the tritone; this specific vector is referred to in this text as the ‘common-practice’ or ‘tonal consonance’ vector. |
| <i>Pitch Scale Hierarchy</i> | The pitch hierarchy (if any) is given in the form of scale tones and a tonic. For instance, a D major scale is given as 2, [0, 2, 4, 5, 7, 9, 11] and an A minor pentatonic scale as 9, [0, 3, 5, 7, 10]. Other more sophisticated encodings of |

¹ The current version of the GCT algorithm requires a symmetric consonance vector for complementary intervals; for instance, if interval 2 is dissonant so is interval 10 (i.e., major second and minor seventh are both dissonant). This means, essentially, that the vector can be shortened to correspond to intervals from 0 (unison) to 6 (tritone). However, in principle, one may adjust the algorithm so as to allow different consonance values for complementary intervals (e.g., major seventh less dissonant than minor second). For this reason, the full 12-interval vector is retained. Additionally, non-binary consonance/dissonance values may be used, allowing for a more refined consonance vector. Instead of filling in the consonance vector with 0s and 1s, it can be filled with fractional values that reflect degrees of consonance derived from perceptual experiments (see, e.g., Hutchinson and Knopoff, 1978) or values that reflect culturally-specific preferences. Such values may improve the algorithm’s performance and resolve some ambiguities in certain cases.

pitch hierarchies are possible, but this simple encoding suffices for the purposes of this study.

Input chord

A list of pitch classes (MIDI pitch numbers modulo 12).

The GCT algorithm, shown in Fig. 2.1, encodes most chord types ‘correctly’ in the standard tonal system. For instance, the note simultaneity [C,D,F \sharp ,A] or [0,2,6,9] in a G major key is interpreted as [7,[0,4,7,10]], i.e., as a dominant seventh chord. Figure 2.2 shows an example of how the GCT algorithm labels a chord.

The algorithm successfully labels most tonal chords (see Sect. 2.3.2 and Cambouropoulos et al., 2014 for examples). However, it is undecided in some cases, and even makes ‘mistakes’ in other cases. In most instances of multiple encodings, it is suggested that these ideally should be resolved by taking into account other harmonic factors (e.g., bass line, harmonic functions, tonal context, etc.). For instance, the algorithm gives two possible encodings for a [0,2,5,9] pc-set, namely minor seventh chord or major chord with added sixth (i.e., [2,[0,3,7,10]] and [5,[0,4,7,9]], respectively); such ambiguity may be resolved if tonal context is taken into account. For the [0,3,4,7] pc-set with root 0, the algorithm produces two answers: a major chord with extension [0,[0,4,7,15]] and a minor chord with extension [0,[0,3,7,16]]. This ambiguity may be resolved if key context is taken into account. For instance, [0,4,7,15] would be selected in a C major or G major context and [0,3,7,16] in a C minor or F minor context. Symmetric chords, such as the augmented chord or the diminished seventh chord, are inherently ambiguous; the algorithm suggests multiple encodings which can be resolved only by taking into account the broader harmonic

Algorithm 1 GCT algorithm (core)

Input: (i) the pitch scale (tonality), (ii) a vector of the intervals considered consonant, (iii) the pitch class set (pc-set) of a note simultaneity

Output: The roots and types of the possible chords describing the simultaneity

- 1: find all maximal subsets of pairwise consonant tones
 - 2: select maximal subsets of maximum length
 - 3: **for** all selected maximal subsets **do**
 - 4: order the pitch classes of each maximal subset in the most compact form (chord ‘base’)
 - 5: add remaining pcs (‘extensions’) above highest element of chord base (add octave, if needed)
 - 6: the lowest tone of the chord is the ‘root’
 - 7: transpose the tones of the chord so that the lowest becomes 0
 - 8: find position of the ‘root’ with respect to the given tonal centre (pitch scale)
 - 9: **end for**
-

Fig. 2.1 The GCT algorithm

Input: G major scale: [7, [0, 2, 4, 5, 7, 9, 11]]
Input: Consonance vector: [1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0]
– Input [50, 60, 66, 69, 74]
– Input converted to pc-set: [0, 2, 6, 9]
– Maximal consonant subset: [2, 6, 9]
– Rewrite in narrowest range: [2, 6, 9]
– Dissonant tone 0 goes to the end (i.e., by adding an octave): [2, 6, 9, 12]
– Lowest tone is root, i.e., 2 (note D)
– Chord in root position: [2, [0, 4, 7, 10]] (i.e., major chord with minor seventh)
– Relative position in key: root is 7 semitones above the tonic G
– Chord in relative position: [7, [0, 4, 7, 10]]
– No other maximal subset exists.
Output: [7, [0, 4, 7, 10]] (i.e., dominant seventh chord)

Fig. 2.2 GCT chord labelling example

context (primarily the next chord). A half-diminished seventh chord [0, 3, 6, 10], is incorrectly encoded as [0, 3, 7, 9], i.e., minor chord with added 6th, since the base chord of the half-diminished chord is a diminished fifth interval (i.e., tritone) which is dissonant and should be avoided. A preliminary evaluation of the GCT algorithm was also carried out on the Kostka–Payne dataset (see Sect. 2.3.2).

Since the aim of this algorithm is not to perform sophisticated harmonic analysis, but rather to find a practical and efficient encoding for tone simultaneities (to be used, for instance, in creative, computer-assisted, harmonic generation that employs combination of harmonic components from diverse idioms—see Sect. 2.3.2), we decided to extend the algorithm by adding the additional steps shown in Fig. 2.3, so that it outputs in every case a single chord type for each simultaneity (no ambiguity).

The additional steps select chord type [2, [0, 3, 7, 10]] over [5, [0, 4, 7, 9]] for the [0, 2, 5, 9] pc-set (see above), as this encoding has maximal overlapping between the two maximal subsets. The maximal overlapping step of the GCT algorithm, in the case of the ‘standard’ tonal consonance vector, amounts to a preference for the ‘stack of thirds’ principle of common tonal chord labelling. The second additional step (if the first step gives more than one option) prefers a chord encoding where non-scale notes are at the end (this rule is not robust and requires further examination). The last step for finding a unique solution is just a simple temporary fix; ideally, the GCT algorithm should be extended to include other factors such as bass note, next chord and, more generally, harmonic context to resolve ambiguity.

2.3.2 Examples of Harmonic Analysis with GCT

An example harmonic analysis of a Bach Chorale phrase illustrates the proposed GCT chord representation (Fig. 2.4). For a tonal context, chord types are optimized

Algorithm 2 GCT Algorithm (additional steps)—for unique encoding


-
- 1: **if** more than one maximal subset **then**
 - 2: Overlapping of maximal subsets: create a sequence of maximal subsets by ordering them so as to have maximal overlapping between them and keep the maximal subset that appears first in the sequence as base of the chord.
 - 3: Avoid non-scale notes in the base: if more than one merged sequence, prefer maximal subset at the base that contains only pcs that appear in the given scale (tonal context).
 - 4: If the above do not give a unique solution, choose the consonant maximal subset that was calculated first as base.
 - 5: **end if**
- Additional adjustment:* for dyads, in a tonal context, prefer perfect fifth over perfect fourth, and prefer seventh to second intervals.
-

Fig. 2.3 The GCT algorithm: additional steps to generate unique encodings

such that pcs at the left-hand side of chords contain only consonant intervals (i.e., thirds and sixths, and perfect fourth and fifths). For instance, the dominant 7th chord is written as $[0, 4, 7, 10]$ since set $[0, 4, 7]$ contains only consonant intervals whereas 10, which introduces dissonances, is placed on the right-hand side—this way the relationship between major chords and dominant seventh chords remains rather transparent and is easily detectable (Kaliakatsos-Papakostas et al., 2015). Within the given D major key context, it is simple to determine the position of a chord type with respect to the tonic. For example, $[7, [0, 4, 7, 10]]$ means a dominant seventh chord whose root is 7 semitones above the tonic. This way we have an encoding that is analogous to the standard Roman numeral encoding (Fig. 2.4, top row). If the tonal context is changed, and we have a chromatic scale context (arbitrary ‘tonic’ is 0, i.e., note C) and we consider all intervals equally ‘consonant’, we get the second GCT analysis in Fig. 2.4, which amounts to normal orders (not prime forms) in a standard pc-set analysis. For tonal music, this pc-set-like analysis is weak as it misses out or obscures important tonal hierarchical relationships. Note that relative ‘roots’ to the ‘tonic’ 0 are preserved as they can be used in harmonic generation tasks (see comment on transposition values in the Messiaen example in Fig. 2.7 below).

Three further examples are presented below that illustrate the application of the GCT algorithm on diverse harmonic textures. The first example (Fig. 2.5) is taken from Beethoven’s *Andante Favori*. In this example, GCT encodes classical harmony in a straightforward manner. All instances of the tonic chord are tagged as 0, $[0, 4, 7]$; the dominant seventh (inverted or not) is 7, $[0, 4, 7, 10]$ and it appears once without the fifth $[7]$; the third-from-last chord is a minor seventh on the second degree encoded as 2, $[0, 3, 7, 10]$; the second and fourth chords are Neapolitan chords

J.S.Bach - Chorale 54 (Lobt Gott, ihr Christen, allzugleich) in G major - 2nd phrase



Roman Numeral Analysis:

D major	I ⁶	vii ⁶ _o	I	ii ⁶	V ⁷	I
---------	----------------	-------------------------------	---	-----------------	----------------	---

GCT Analysis (tonal major profile)

2,[0,2,4,5,7,9,11]	0,[0,4,7]	11,[0,3,6]	0,[0,4,7]	2,[0,3,7]	7,[0,4,7,10]	0,[0,4,7]
--------------------	-----------	------------	-----------	-----------	--------------	-----------

Pc-Set Analysis (chromatic scale):

normal orders	[0,4,7]	[0,3,6]	[0,4,7]	[0,3,7]	[0,2,6,9]	[0,4,7]
prime forms	[0,3,7]	[0,3,6]	[0,3,7]	[0,3,7]	[0,3,6,8]	[0,3,7]

GCT Analysis (atonal profile)

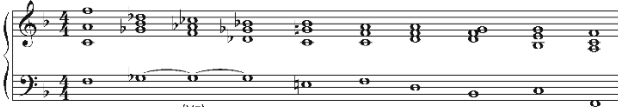
[0,1,2,3,4,5,6,7,8,9,10,11]	2,[0,4,7]	1,[0,3,6]	0,[0,4,7]	4,[0,3,7]	7,[0,2,6,9]	2,[0,4,7]
-----------------------------	-----------	-----------	-----------	-----------	-------------	-----------

Fig. 2.4 Chord analysis of a Bach chorale phrase by means of traditional Roman numeral analysis, pc-sets and two versions of the GCT algorithm

encoded as 1, [0,4,7] (which means a major chord on the lowered second degree) with a secondary dominant in between.

The second example, in an extended chromatic style, is from Richard Strauss' *Till Eulenspiegel*. Piston (1978) provides a tentative partial analysis (Fig. 2.6). He suggests that "any generalized appearance of tonality breaks down under the weight of the rapid harmonic rhythm and distantly related harmonic functions" (p. 522). The GCT algorithm was applied to this excerpt for the standard tonal consonance vector, and for the key of F major (5, [0,2,4,5,7,9,11]). The algorithm assigns chord types correctly to all chords (in agreement with Piston's analysis) except for the seventh chord, which is a symmetric diminished seventh, for which the algorithm selects the wrong root. Additionally, the algorithm assigns chord types to all chords that have been left unanalysed by Piston. Of course, excluding some chords is an integral part of the analytic process (the analyst decides which chords are structurally important and should be labelled); the algorithm is not sophisticated enough to make such decisions. However, the chord types the GCT algorithm assigns to these chords are not arbitrary and may be potentially useful for an automated harmonic reasoning system.

In the third example, from a piece by Messiaen (Fig. 2.7), the second half of the excerpt is a transposed repetition (lowered by three semitones) of the first half.



I II^N (V⁷) II^N V⁶⁵ I vi ii⁶⁵ V⁷ I

[0,[0,4,7]] [1,[0,4,7]] [0,[0,3,6]] [1,[0,4,7]] [7,[0,4,7,10]] [0,[0,4,7]] [9,[0,3,7]] [2,[0,3,7,10]] [7,[0,4,7,10]] [0,[0,4,7]]

Fig. 2.5 Reduction of bars 189–198 of Beethoven's *Andante Favori*. Top row: manual Roman numeral harmonic analysis; bottom row: GCT analysis. GCT successfully encodes all chords, including the Neapolitan sixth chord (the pedal G flat note in the third chord is omitted)

W. Piston F: 1 VI V2/V III V7/VI IV -1165 III6 ? ? ? ? IV64 V1/V V2
 GCT 0.047 8.047 2.04710 3.047 4.04710 5.047 9.0369 4.037 10.04710 6.037 7.03710 8.037 5.047 1.047 7.047

Fig. 2.6 Bars 1–3 of R. Strauss’ *Till Eulenspiegel*. Partial harmonic analysis by W. Piston and chord types determined by the GCT algorithm (condensed GCT encoding—without commas)

The GCT can be used to capture transposition-invariant matching, if the ‘roots’ of chords are replaced by intervals between ‘roots’. This way the representation becomes transposition-invariant. In this instance, the interval pattern between ‘roots’ of the first and second parts is $[+6, -3, +3, +3, -3]$ for the ‘tonal’ version, and $[-4, -3, +3, -2, -3]$ for the ‘atonal’ version. This is a reason for preserving the ‘roots’ (relative to an arbitrary reference pitch class: pc 0) in the atonal version. In pitch class set theory, normal orders do not have ‘roots’; however, they have transposition values (T0–T11) in relation to a reference pc (normally pc 0). The normal orders with transposition values of pc-set theory are equivalent to the GCT for the ‘atonal’ consonance vector.

We tested the GCT algorithm on the Kostka–Payne dataset (Temperley, 2001b). This dataset consists of the 46 excerpts that are longer than 8 measures from the workbook accompanying Kostka and Payne’s (1995) theory textbook. Given the local tonality (key), the GCT algorithm was applied to all the Kostka–Payne excerpts. Then, the resulting GCTs were compared to the Kostka–Payne ground truth (i.e., the Roman numeral analysis not taking into account inversions). From the 919 chords of the dataset, GCT encodes fully in compliance with the human analysis 846 chords, and 72 chords are labelled differently. This means that the algorithm labels correctly 92% of all the chords. Twenty-three mislabelled chords were diminished seventh chords—these symmetric chords can have as their root any of the four constituent notes; twenty-two half-diminished chords $[0, 3, 6, 10]$ were labelled as minor chords with added sixth $[0, 3, 7, 9]$; seventeen cases had a salient note missing (e.g., diminished chord without root, dominant seventh without third, half-diminished seventh without

GCT-common
 practice conso- 0,[0,4,10,13] 3,[0,4,7,18] 9,[0,3,7,10] 9,[0,4,10,13] 0,[0,4,7,18] 6,[0,3,7,10]
 nance vector 6,[0,4,7,18] 6,[0,4,7,18] 6,[0,3,7,10] 3,[0,4,7,18] 3,[0,4,7,18] 3,[0,3,7,10]
 GCT-atonal 10,[0,2,3,6] 3,[0,4,6,7] 4,[0,3,5,8] 7,[0,2,3,6] 0,[0,4,6,7] 1,[0,3,5,8]
 cons. vector 6,[0,4,6,7] 6,[0,4,6,7] 1,[0,3,5,8] 3,[0,4,6,7] 3,[0,4,6,7] 10,[0,3,5,8]
 (normal order)

Fig. 2.7 Reduction of the first six bars of O. Messiaen’s *Quartet for the End of Time*, *Quartet VII*. The piece is based on the octatonic scale: $[0, 1, 3, 4, 6, 7, 9, 10]$. Top row: GCT encoding for standard common-practice consonance vector; bottom row: GCT encoding for atonal harmony—all intervals ‘consonant’ (this amounts to pc-set ‘normal orders’). See text for further details

third) and this resulted in finding a wrong root; eight chords were misspelled because they appeared over a pedal note (the pedal note was included in the chord); two sus4 chords [0, 5, 7] were written incorrectly as [0, 5, 10] (e.g., [C,F,G] contains the dissonant interval [F,G] and is erroneously re-ordered as [G,C,F]).

In the context of tonal music, for the standard tonal consonance vector, the GCT algorithm makes primarily the following types of mistake: first, it is undecided with regard to the root of symmetric chords such as diminished seventh chords and augmented triads; second, it assigns the wrong root to chords that have ‘dissonant’ intervals at their base, such as diminished fifths in half-diminished chords or major second in sus4 chords; and, finally, tertian chords that have notes missing from their base (e.g., missing third in seventh chords) are misinterpreted as their upper denser part is taken as the chord’s base and the lower root as an extension.

In order to correct such cases, a more sophisticated model for harmonic analysis is required. Such a model should take into account voicing (e.g., the bass note), chord functions (see Kaliakatsos-Papakostas et al.’s (2015) proposal for organizing GCTs into functional tonal categories), and, even, higher-level domain-specific harmonic knowledge (e.g., specific types of chords used in particular idioms). However, the aim of the current proposal is not a tool for tonal analysis, but a general chord representation that can be applied to different harmonic systems (including the tonal system).

The GCT algorithm captures reasonably well the common-practice Roman-numeral harmonic analysis encoding scheme (for the ‘standard’ consonance vector). Additionally, it adapts to non-tonal systems, such as atonal, octatonic or traditional polyphonic music. The question is whether the GCT representation works well on such non-tonal systems. The GCT representation has been employed in the case of traditional polyphonic music from Epirus (Kaliakatsos-Papakostas et al., 2014). In this study, song transcriptions were first converted to the GCT encoding, then a learning HMM scheme was employed to learn chord transitions and, finally, this knowledge was used to create new harmonizations in the polyphonic style of Epirus. Ongoing research is currently studying the application of GCT to various harmonic idioms, from mediaeval music to 20th century music, and various pop and folk traditions.

What might such ‘universal’ harmonic encoding systems be useful for? Apart from music-theoretic interest and cognitive considerations and implications, a general chord encoding representation may allow generic harmonic systems to be developed that may be adapted to diverse harmonic idioms, rather than designing ad hoc systems for individual harmonic spaces. This was the primary aim for devising the General Chord Type (GCT) representation. In the case of the project COINVENT (Schorlemmer et al., 2014) a creative melodic harmonization system is required that relies on conceptual blending between diverse harmonic spaces in order to generate novel harmonic constructions; mapping between such different spaces is facilitated when the shared generic space is defined with clarity, its generic concepts are expressed in a general and idiom-independent manner, and a common general representation is available (Kaliakatsos-Papakostas and Cambouropoulos, 2014).

Overall, the GCT representation and algorithm are an attempt to create a flexible and adaptable representation, capable of encoding different harmonic idioms. Clearly, the algorithm can be extended and improved in different ways, depending on the task at hand. Using it, for instance, for learning chord transition probabilities from different corpora, unambiguous results are handy (each chord receiving a unique encoding). On the other hand, if one wants to study human harmonic analysis or perception, it may be useful to retain ambiguity and compare results with human empirical data. The GCT representation attempts to capture commonly used properties of chords (root, basic type, extension) in diverse styles (giving, however, a privileged vantage point to hierarchic tonal systems). Various refinements of the algorithm are possible depending on the context of the particular music-analytic or creative task to which it is being applied.

2.4 Pitch Class Chord Transition Representation

Is it possible to devise a chord sequence representation that ‘captures’ the intervallic content of chord transitions in a transposition-invariant and idiom-independent way without recourse to the concept of a chord? Is it possible to define a harmonic equivalent to the idiom-independent transposition-invariant ‘pitch interval’? In this section a proposal is made for a chord transition representation that encodes chord *transitions* independently of the absolute pitches of actual chords, or any other representation of the notion of a chord.

Chords are usually represented either as collections of pitch-related values (e.g., note names, MIDI pitch numbers, pitch class sets, chroma vectors, etc.) or as chord root transitions within a given tonality following traditional harmonic analysis (e.g., Roman numeral analysis, guitar chords, etc.). In the case of an absolute pitch representation (such as chroma vectors, i.e., pitch class profiles) transpositions are not accounted for (e.g., twelve transpositions of a given query are necessary to find all possible occurrences of the query in a dataset). On the other hand, if harmonic analytic models are used to derive harmonic descriptions of pieces (e.g., chords as degrees within keys or tonal functions), more sophisticated processing is possible; in this case, however, models rely on complicated harmonic analytic systems, and, additionally, are limited to the tonal idiom.

One obvious way to represent chord sequences in a transposition-invariant and idiom-independent way is to encode chord transitions as intervals between the first pitch classes in the prime forms of the transpositional equivalence classes of the pitch class sets of two chords (e.g., for transition C–F: [5, [0, 4, 7], [0, 4, 7]]). This works well but it relies on the conversion of chords into pitch class sets (prime forms)—that is, it requires encoding the two chords in some abstract form which implies various assumptions and processing steps (see discussion in Sect. 2.3.2 above, relating to Fig. 2.7, for a GCT example in the same vein). Another possible way is to employ voice-leading intervals following, for instance, Tymoczko’s (2011) pitch-class voice-leading formalism. For example, for transition C–F, the voice-leading is represented

by $(C, E, G) \xrightarrow{0,1,2} (C, F, A)$. This representation relies on coupling the first chord (i.e., the notion of chord is necessary) with voice-leading paths describing thus a chord transition; this formalism gets more complicated when successive chords do not contain the same number of notes, as this results in a non-bijective relation between pitch sets. All of the above encoding schemes require encoding one or both chords delimiting a chord transition. Is it possible to represent a chord transition solely in terms of pitch intervals? The proposed DIC vector representation provides one possible solution. This can be seen as a music-theoretic exercise per se, or as a practical encoding scheme that may be useful for certain tasks.

Most computational models of harmonic processing rely on some representation of individual chords. There are very few attempts, however, to represent chord transitions. For instance, de Haas et al. (2008, 2013) represent chord transitions as chord distance values, adapting a distance metric from Lerdahl’s tonal pitch space (Lerdahl, 2001). This representation is geared towards the tonal system; additionally, a chord transition being represented by a single distance value may capture certain important properties of the transition but may abstract out other important information.

2.4.1 The Directed Interval Class (DIC) Vector Representation

A proposal for representing chord transitions in an idiom-independent manner is presented in this section. A harmonic transition between two chords can be represented by a *Directed Interval Class* (DIC) vector (Cambouropoulos, 2012; Cambouropoulos et al., 2013). This representation allows the encoding of chord transitions at a level higher than individual notes that is transposition-invariant and idiom-independent (analogous to pitch intervals that represent transitions between notes).

The proposed 12-dimensional vector is an adaptation of Lewin’s (1959) interval function (see also Lewin, 2001, 2007). Lewin’s function encodes the frequencies of occurrence of the pitch intervals (from 0 to 11) that occur between the notes in one pitch class set and the notes in another. In the current proposal, Lewin’s function is modified such that the proposed vector encodes all pitch intervals (from 0 to 6 including $+/-$ for direction; 0 and 6 are always positive) between all the pairs of notes of two successive chords. The DIC vector is equivalent to Lewin’s interval function; it is simply a re-ordering of Lewin’s intervals using directed pitch interval classes. The proposed representation attempts to capture the fact that voice-leading practice involves mostly small intervals (e.g., unison, step-wise motion) connecting notes from one chord to another (cf. Tymoczko’s (2011) pitch-class voice leading). From the left-hand side to the right of the proposed interval vector, we move from unison to small (up or down) intervals up to the tritone. The DIC vector simply makes the ‘voice leading’ character of a chord transition more apparent. As will be proposed toward the end of this section, it may be possible to shorten the proposed vector by keeping only the smaller intervals at the left-hand side of the vector without losing its expressivity.

The 12-dimensional DIC vector features the following directed interval classes in its twelve positions: 0 (unison), +1, -1, +2, -2, +3, -3, +4, -4, +5, -5, 6 (tritone). The reason for opting for this ordering is that the absolute size of intervals increases from left to right, and this places small intervals, that are thought to be most strongly connected with voice-leading, at the ‘privileged’ beginning of the vector. The proposed vector encodes the frequency of occurrence of all directional interval classes between all the pairs of notes of two successive chords. That is, from each note of the first chord, all intervals to all the notes of the second chord are calculated. Direction of intervals is preserved (+, -), except for the unison (0) and the tritone (6) that are undirected. Interval size takes values from 0 to 6 (interval class). If an interval *X* is greater than 6, then its complement $12 - X$ in the opposite direction is retained (e.g., ascending minor seventh ‘+10’ is replaced by its equivalent complement descending major second ‘-2’).

As an example, the transition vector for the progression I–V is given by the DIC vector: $Q = \langle 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 3, 0 \rangle$ (which means: 1 unison, 0 ascending minor seconds, 1 descending minor second, 1 ascending major second, etc.)—see Fig. 2.8, and further examples in Fig. 2.9.

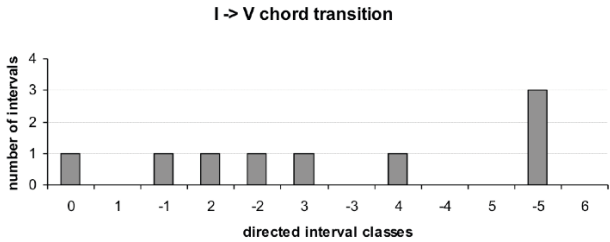


Fig. 2.8 The DIC vector, $\langle 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 3, 0 \rangle$, for the chord transition I–V, depicted as a bar graph

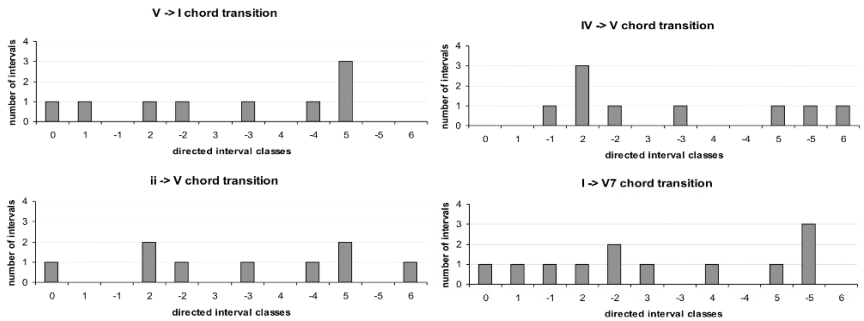


Fig. 2.9 DIC vectors for four standard tonal chord transitions: V–I, IV–V, ii–V, I–V7

For a given harmonic (e.g., tonal) context, the DIC vector is unique for many chord transitions. However, there are a number of cases where different tonal transitions have the same vector. For instance, the transitions I–V and IV–I share the same DIC vector as their directed interval content is the same; it should be noted, that, heard in isolation (without a tonal centre reference), a human listener cannot tell the difference between these two transitions. Another case is when one of the two chords is symmetric (e.g., an augmented triad or diminished seventh chord); this is actually an interesting case that agrees with music theory and intuition in the sense that, say, diminished seventh chords are considered ambiguous and can resolve to different chords leading to different tonal regions or keys.

In addition to the above cases of ambiguity (that are musically meaningful), the DIC vector encodes identically a chord transition and its retrograde inversion, i.e., any specific chord transition has the same DIC vector as its retrograde inversion. For instance, the retrograde inversion of a *major triad* progressing to a *major triad* by an interval *X* is a *minor triad* progressing to a *minor triad* by the same interval *X*; these two transitions share the same DIC vector. This is an inherent property of the DIC vector which reduces its descriptive power, and may have serious ramifications for certain tasks (see below).

Cambouropoulos et al. (2013) evaluated the proposed DIC representation on a harmonic recognition task, in which the accuracy of recognition of harmonic queries was tested in a small database of chord sequences (harmonic reductions) derived from diverse musical idioms and styles. More specifically, standard chord progressions were included from Bach chorales, along with harmonic progressions from modal Greek *rebetiko* songs, polyphonic songs from Epirus, Beatles songs and non-tonal pieces by Béla Bartók, Olivier Messiaen, Claude Debussy, and Erik Satie (31 chord reductions of pieces, collectively containing 957 chords in all). Both the query sequence and the chord progressions in the dataset were converted to DIC vectors and exact matching for recognition was employed (approximate matching was also considered using wildcards—see below). The results obtained with the algorithm were judged by human music analysis experts. For this small dataset, relatively longer sequences consisting of four or more chords were uniquely identified in the correct positions in the pieces where they originated. For instance, we examined exhaustively queries drawn from J. S. Bach’s chorale, “Ein feste Burg ist unser Gott” (BWV 302), consisting of three or four chords; the longest sequence found in at least one other piece was a 4-chord sequence (the first four chords identified in position 26 of *Strawberry Fields* by The Beatles). Obviously, if the dataset is significantly extended we expect to find more occurrences of relatively longer harmonic queries. Many examples of harmonic queries and matches are reported and discussed by Cambouropoulos et al. (2013).

Overall, the harmonic recognition model behaves as expected, and has sufficient distinctive power to discern the harmonic individualities of these different harmonic languages. Almost all of the harmonic queries were correctly detected (see one problem below) and all queries containing at least three chords were identified without mistakes. Apparently the model is capable of finding repeating harmonic patterns even though pieces are in different keys (transposition invariance). Additionally,

recall that the system has no knowledge of the different kinds of harmonic systems (tonal, modal, chromatic, atonal, etc.), and it is therefore interesting that it detects correctly any kind of harmonic query in diverse harmonic idioms.

Interestingly, the harmonic recognition model is equally successful and accurate when only the first five vector components are used. We tested all the above queries using only the first five vector entries $[0, 1, -1, 2, -2]$ (out of the 12) which correspond to unison (i.e., common pitches between two chords) and steps (i.e., ascending and descending semitones and tones); the resulting matches were the same in every case. These small intervals may be thought of as being mostly related to voice-leading as it is standard practice to try to connect chords avoiding larger intervals (using common notes and step movements). The reduction of the DIC vector to a 5-component subvector, enhances the cognitive plausibility of the proposed representation. Even though no cognitive claims are made in this chapter, we just mention that representing the transition between two chords as the small intervals that link adjacent pitches (being potentially part of individual harmonic voices) affords this representation potential cognitive validity. Finally, a mathematical analysis of the redundancy contained in the full DIC vector is needed; it is likely that, for example, for tertian chords, the larger pitch intervals can be recovered from the smaller ones. Both cognitive and mathematical properties of such a representation should be explored further in future studies. In any case, this reduced vector results in better computational efficiency.

As mentioned above, a chord sequence shares a DIC vector with its retrograde inversion. In the case of harmonic matching, this is not a serious problem if the sequences sought are relatively long (say, at least 3 chords). The reason is that the additional context of neighbouring chords often disambiguates the overall sequence. For instance, a query of two major chords an ascending fourth apart returns one hundred instances in our small dataset, some of which erroneously consist of two minor chords an ascending fourth apart; if, however, our query is preceded by a major chord one tone lower than the first chord, then we find only 12 instances of the sequence (corresponding to IV–V–I). In another example, the transition between two major chords an ascending tone apart returned 23 instances that correspond most likely to a IV–V chord progression (even though it may correspond to a ii–iii transition between minor chords). If the first chord is preceded by a diminished chord a semitone lower, then the whole sequence of three chords is found only once (in chorale BWV 302—the sequence is vii^o–I–V/V). The specific context restrains the search drastically. Longer sequences are more unambiguous.

Finally, an important issue not explored sufficiently in this study is chord progression similarity, i.e., how similar two chord sequences are. As it stands, a IV–I transition and a V7–I transition are different (not matched) because their DIC vectors are not identical. Similarly, a V–I transition is not matched to another V–I transition if a note is missing such as the fifth of the first or second chord. Such relations can be captured if certain tolerances are allowed (approximate matching). For instance, if all entries of one vector are smaller than the corresponding entries of the other vector, and the sum of the differences is three or less, then sequences such as V–I and V7–I would be matched. This is a kind of approximate matching where the difference between individual DIC vector entries is less than or equal to δ and the

sum of all differences less than γ (see Cambouropoulos et al. (2002) for another study of (δ, γ) -matching in music). The similarity relations between vectors is an open issue for further research.

In the current implementation, wild cards can be inserted in order to allow a certain tolerance in the matching. Disabling entries 6–12 in the vector, i.e., using only the first five components, did not seem to make a difference as mentioned earlier in this section. Trying out a more radical example, we queried the system with the vector $\langle 0, *, *, *, *, *, *, *, *, *, *, 2 \rangle$; this means we are looking for a chord succession that contains no common notes but in which there exist two distinct tritone relations between different notes of the chords. The system returned 7 instances: 4 in the rebetiko songs; one in *Michelle* by The Beatles that corresponds to the transition of major chord to minor chord a tone lower (or the reverse); one in Bartók's fourth *Romanian Folk Dance* that corresponds to the transition of minor chord 7th to major chord a tone higher; and one in the first measures of Debussy's *Nuages* that corresponds to the transition of a perfect fourth harmonic interval to a perfect fifth a semitone lower. Such experiments allow the investigation of broader similarity relations.

Cambouropoulos (2012) attempted to test the effectiveness of the DIC vector representation in a harmonic progression similarity task. The main assumption there was that, if this representation encodes aspects of the harmonic content of chord progressions sufficiently well, then, given an appropriate distance metric, similarity between different chord progressions can be calculated and chord progressions can be clustered together in meaningful classes. We use the DIC vector representation as a basis for calculating the distance between simple chord progressions in two preliminary tests. In one case, we calculate the distances between 12 simple tonal triadic progressions; whereas in the second case, we have 11 jazz progressions. For this preliminary testing, we employed a very simple distance metric between the DIC vector sequences, namely the city block distance. In this initial experiment we employed phylogenetic trees (branching diagrams) to visualize the distance or similarity relations between the given chord sequences.

In the first preliminary test, we constructed a set of twelve triadic tonal chord progressions, each consisting of four chords (Fig. 2.10). The twelve chord progressions were constructed such that they were organized into three groups of four instances each, corresponding to the following harmonic function progressions: T–S–D–T, T–S–T–D, and S–T–D–T (T=tonic, S=subdominant, D=dominant). The simple similarity method employed in this study resulted in a phylogenetic tree branching that splits the twelve progressions into three groups that correspond with the expected harmonic function progressions (Fig. 2.10). This very simple method manages to group together successfully these chord progressions without any knowledge of tonality, keys, chord roots, scales or any other sophisticated harmonic concepts. Even if we transpose these progressions to various keys the proposed method would give exactly the same result. The mere intervallic content of these progressions is sufficient for finding similarities between them and organizing them into groups. It should be noted, however, that, if in this example more extensive substitutions of chords are introduced, the resulting tree is less successful, possibly because the distance metric

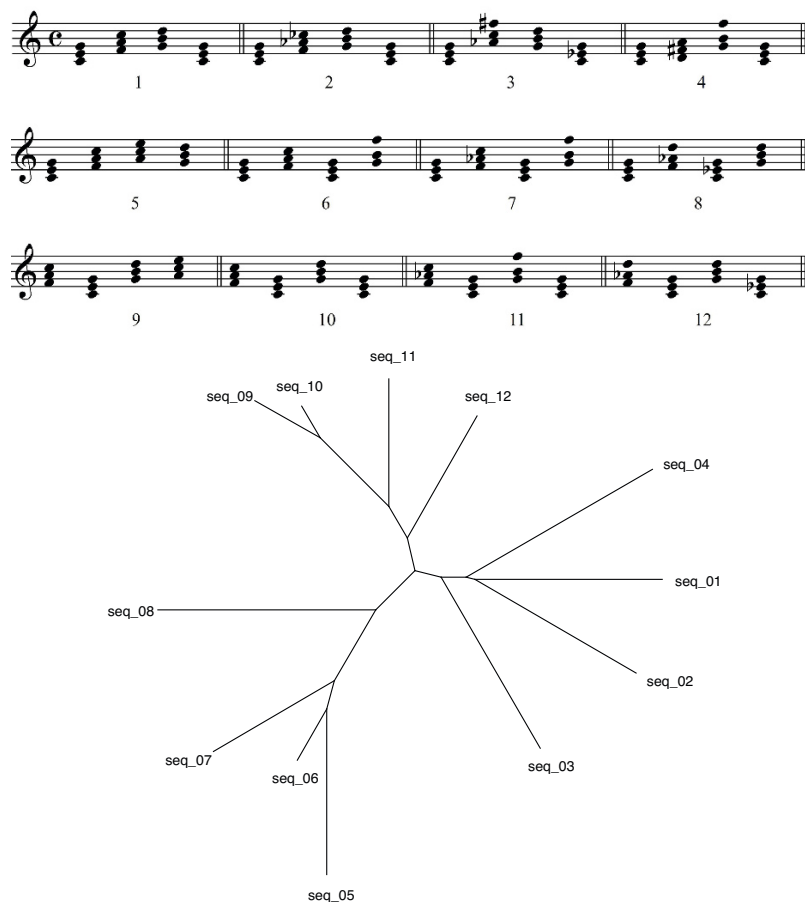


Fig. 2.10 The triadic chord progressions are organized into a phylogenetic tree that illustrates their similarities and grouping based on their DIC vector distances

is extremely elementary. The similarity relations between vectors is an open issue for further research.

In a second preliminary test, we asked an experienced jazz piano performer to write down some jazz chord progressions (same length) and, also, to let us know how she thought they related to each other. The jazz pianist prepared eleven jazz chord progressions, each consisting of 4 chords. As in the previous test, these were organized into an 11×11 distance matrix and, then, a phylogenetic tree was constructed (Fig. 2.11). The jazz musician examined the resulting phylogenetic tree and gave the following feedback:

I think it is very nice and I agree with the main parts. The point on which I would disagree is placing 6 far away from 5, 7, 8; I would place them in [the] same class. Secondly, group 1, 3 is closer to 2 in my opinion; the rest of the tree is very convincing.

The harmonic similarity between these jazz chord progressions seems to be captured reasonably well, despite the simplicity of the proposed model and its total ignorance of jazz harmony. This is encouraging; however, more systematic research is necessary to improve the model and to test it more extensively (e.g., empirical data for the 11 progressions could be gathered from a larger number of jazz musicians).

One might argue that all the above harmonic recognition and classification tasks can be modelled equally well by employing a simple pc-set-based representation (e.g., chord transitions as intervals between the first pitch classes of the transpositional pitch class sets of two chords). This is true. Why, then, use the DIC vector representation at all? First, the DIC vector is very simple to compute (easier than normal orders and prime forms of pc-set theory). Second, it may allow more general distance metrics to be developed for the calculation of distances between chord sequences. Third,

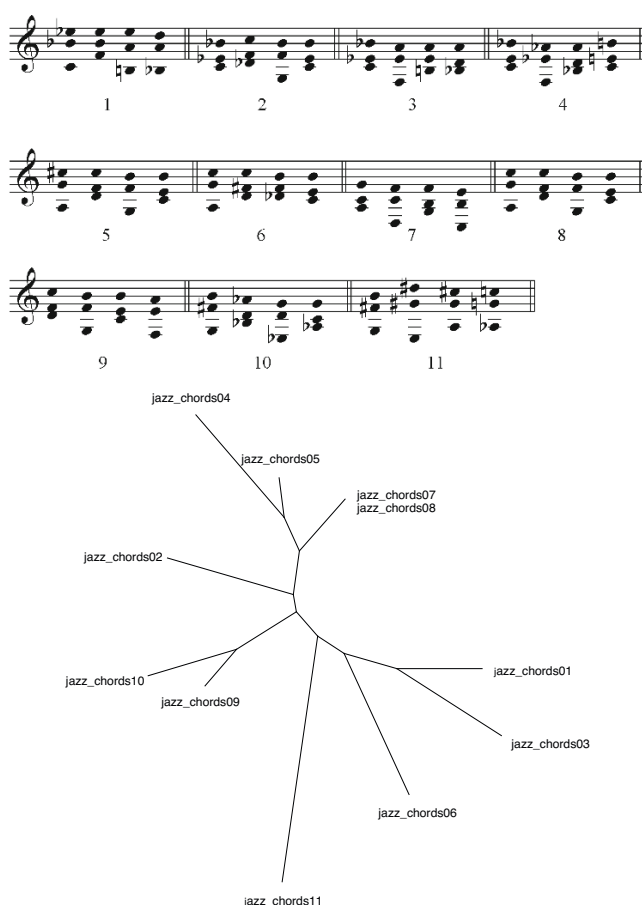


Fig. 2.11 The jazz chord progressions are organized into a phylogenetic tree that illustrates their similarities and grouping based on their DIC vector distances. See text for more details

it can be applied even in cases where it is not straightforward to compute prime forms. For example, it is not straightforward to convert chroma vectors extracted from audio into pitch class sets so that transitions between them may be represented in a transposition-invariant manner; whereas successive chroma vectors can readily be converted into weighted DIC vectors. Finally, the DIC vector (especially some abbreviated form containing, for instance, only small intervals) may afford cognitive relevance in the sense that, in some cases at least, listeners may abstract and categorize directly intervallic content of chord progressions (instead of applying more advanced processing that involves identifying pc-set classes and intervals between them). Further research is necessary to substantiate such claims. Herein only preliminary hints are given as to the potential of the proposed representation.

2.5 Conclusions

This chapter addresses issues of harmonic representation at the level of the musical surface. It is argued that selecting an appropriate representation for chords is crucial for encoding aspects of the musical surface that have perceptual pertinence, and at the same time is paramount for building efficient and meaningful computational models for music analysis. In the course of the chapter, two novel general chord representations were presented and their potential was discussed. The first, the GCT representation, generalizes the standard Roman-numeral representation in such a way as to apply to any idiom; whereas the second, the DIC vector, captures the intervallic content of a transition between two chords. The former is algorithmically more complex but embodies more structured harmonic information (consonance/dissonance, ‘root’, chord type, position in scale) and is thus adequate for music analytic/synthetic tasks in more sophisticated hierarchical musical idioms. It is suggested that both of the proposed chord representations may be used for representing harmonic relations in music from diverse musical idioms (i.e., they are not confined to tonal music) and, therefore, may provide a most appropriate framework for harmonic processing in the domain of computational musicology.

Acknowledgements The project COINVENT acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 611553. Special thanks are due to Costas Tsougras for the preliminary analysis of the Kostka–Payne dataset and for the harmonic reduction examples, Maximos Kaliakatsos-Papakostas for his practical support in preparing this manuscript, and two anonymous reviewers and David Meredith for providing most interesting suggestions on an earlier version of this chapter.

References

- Bregman, A. S. (1994). *Auditory Scene Analysis: The Perceptual Organization of Sound*. MIT Press.
- Burns, E. M. (1999). Intervals, scales and tuning. In Deutsch, D., editor, *The Psychology of Music*, pages 215–264. Academic Press, second edition.
- Cambouropoulos, E. (2008). Voice and stream: Perceptual and computational modeling of voice separation. *Music Perception*, 26(1):75–94.
- Cambouropoulos, E. (2010). The musical surface: Challenging basic assumptions. *Musicae Scientiae*, 14(2):131–147.
- Cambouropoulos, E. (2012). A directional interval class representation of chord transitions. In *Proceedings of the Joint 12th International Conference for Music Perception and Cognition & 8th Conference of the European Society for the Cognitive Sciences of Music (ICMPC-ESCOM 2012)*, Thessaloniki, Greece.
- Cambouropoulos, E., Crochemore, M., Iliopoulos, C. S., Mouchard, L., and Pinzon, Y. J. (2002). Algorithms for computing approximate repetitions in musical sequences. *International Journal of Computer Mathematics*, 79(11):1135–1148.
- Cambouropoulos, E., Kaliakatsos-Papakostas, M., and Tsougras, C. (2014). An idiom-independent representation of chords for computational music analysis and generation. In *Proceeding of the Joint 11th Sound and Music Computing Conference (SMC) and 40th International Computer Music Conference (ICMC)*, ICMC–SMC 2014, Athens, Greece.
- Cambouropoulos, E., Katsiavalos, A., and Tsougras, C. (2013). Idiom-independent harmonic pattern recognition based on a novel chord transition representation. In *Proceedings of the 3rd International Workshop on Folk Music Analysis (FMA2013)*, Amsterdam, Netherlands.
- Cemgil, A. T., Kappen, H. J., and Barber, D. (2006). A generative model for music transcription. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2):679–694.
- de Haas, W. B., Veltkamp, R. C., and Wiering, F. (2008). Tonal pitch step distance: a similarity measure for chord progressions. In *9th International Conference on Music Information Retrieval (ISMIR 2008)*, pages 51–56, Philadelphia, PA.
- de Haas, W. B., Wiering, F., and Veltkamp, R. C. (2013). A geometrical distance measure for determining the similarity of musical harmony. *International Journal of Multimedia Information Retrieval*, 2(3):189–202.
- Deutsch, D. (2012). The processing of pitch combinations. In Deutsch, D., editor, *The Psychology of Music*, pages 249–326. Academic Press, third edition.
- Forte, A. (1973). *The Structure of Atonal Music*. Yale University Press.
- Handel, S. (1989). *Listening: An Introduction to the Perception of Auditory Events*. MIT Press.
- Harte, C., Sandler, M., Abdallah, S. A., and Gómez, E. (2005). Symbolic representation of musical chords: A proposed syntax for text annotations. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, pages 66–71, London, UK.

- Hubbard, T. L. and Datterri, D. L. (2001). Recognizing the component tones of a major chord. *The American Journal of Psychology*, 114(4):569–589.
- Huron, D. (2001). Tone and voice: A derivation of the rules of voice-leading from perceptual principles. *Music Perception*, 19(1):1–64.
- Hutchinson, W. and Knopoff, L. (1978). The acoustic component of Western consonance. *Interface*, 7(1):1–29.
- Jackendoff, R. (1987). *Consciousness and the Computational Mind*. MIT Press.
- Jackendoff, R. and Lerdahl, F. (2006). The capacity for music: What is it, and what's special about it? *Cognition*, 100(1):33–72.
- Jordanous, A. (2008). Voice separation in polyphonic music: A data-driven approach. In *Proceedings of the International Computer Music Conference 2008*, Belfast, UK.
- Kaliakatsos-Papakostas, M. and Cambouropoulos, E. (2014). Probabilistic harmonisation with fixed intermediate chord constraints. In *Proceedings of the Joint 11th Sound and Music Computing Conference (SMC) and 40th International Computer Music Conference (ICMC)*, Athens, Greece.
- Kaliakatsos-Papakostas, M., Katsiavalos, A., Tsougras, C., and Cambouropoulos, E. (2014). Harmony in the polyphonic songs of Epirus: Representation, statistical analysis and generation. In *4th International Workshop on Folk Music Analysis (FMA 2014)*, Istanbul, Turkey.
- Kaliakatsos-Papakostas, M., Zacharakis, A., Tsougras, C., and Cambouropoulos, E. (2015). Evaluating the General Chord Type algorithm in tonal music and organising its output in higher-level functional chord categories. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, Malaga, Spain.
- Kostka, S. and Payne, D. (1995). *Tonal Harmony*. McGraw-Hill.
- Laitz, S. G. (2012). *The Complete Musician: An Integrated Approach to Tonal Theory, Analysis, and Listening*. Oxford University Press.
- Lerdahl, F. (2001). *Tonal Pitch Space*. Oxford University Press.
- Lerdahl, F. and Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. MIT Press.
- Lewin, D. (1959). Intervallic relations between two collections of notes. *Journal of Music Theory*, 3(2):298–301.
- Lewin, D. (2001). Special cases of the interval function between pitch-class sets x and y . *Journal of Music Theory*, 45(1):1–29.
- Lewin, D. (2007). *Generalized Musical Intervals and Transformations*. Oxford University Press.
- Locke, S. and Kellar, L. (1973). Categorical perception in a non-linguistic mode. *Cortex*, 9(4):355–369.
- Mauch, M., Cannam, C., Davies, M., Dixon, S., Harte, C., Kolozali, S., Tidhar, D., and Sandler, M. (2010). OMRAS2 metadata project 2009. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, Utrecht, The Netherlands.
- Parncutt, R. (1989). *Harmony: A Psychoacoustical Approach*. Springer.

- Parncutt, R. (1994). Template-matching models of musical pitch and rhythm perception. *Journal of New Music Research*, 23(2):145–167.
- Parncutt, R. (1997). A model of the perceptual root(s) of a chord accounting for voicing and prevailing tonality. In Leman, M., editor, *Music, Gestalt, and Computing*, volume 1317 of *Lecture Notes in Computer Science*, pages 181–199. Springer.
- Piston, W. (1978). *Harmony*. Norton. Revised and expanded by M. DeVoto.
- Povel, D.-J. and Jansen, E. (2001). Perceptual mechanisms in music processing. *Music Perception*, 19(2):169–197.
- Ryynänen, M. P. and Klapuri, A. P. (2008). Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32(3):72–86.
- Schorlemmer, M., Smaill, A., Kühnberger, K.-U., Kutz, O., Colton, S., Cambouropoulos, E., and Pease, A. (2014). COINVENT: Towards a computational concept invention theory. In *5th International Conference on Computational Creativity (ICCC) 2014*, Ljubljana, Slovenia.
- Sloboda, J. A. (1985). *The Musical Mind*. Oxford University Press.
- Smith, J. D., Nelson, D. G., Grohskopf, L. A., and Appleton, T. (1994). What child is this? What interval was that? Familiar tunes and music perception in novice listeners. *Cognition*, 52(1):23–54.
- Temperley, D. (2001a). *The Cognition of Basic Musical Structures*. MIT Press.
- Temperley, D. (2001b). Kostka–Payne dataset. Available online at <http://theory.esm.rochester.edu/temperley/kp-stats/>. Last accessed 5 September 2015.
- Temperley, D. (2012). Computational models of music cognition. In Deutsch, D., editor, *The Psychology of Music*, pages 327–368. Academic Press, third edition.
- Tymoczko, D. (2011). *A Geometry of Music: Harmony and Counterpoint in the Extended Common Practice*. Oxford University Press.
- Vernon, P. E. (1934). Auditory perception. I. The Gestalt approach. II. The evolutionary approach. *British Journal of Psychology*, 25:123–139, 265–283.



<http://www.springer.com/978-3-319-25929-1>

Computational Music Analysis

Meredith, D. (Ed.)

2016, XV, 480 p., Hardcover

ISBN: 978-3-319-25929-1