Table of Contents

[Introduction 2](#_Toc192022978)

[Related work 2](#_Toc192022979)

[Hindemith’s *Craft of Musical Composition* 2](#_Toc192022980)

[The *ontocomc* ontology 9](#_Toc192022981)

[Class *:PitchClass* 9](#_Toc192022982)

[Class *:Interval* and taxonomy 10](#_Toc192022983)

[Class :C*hord* and taxonomy 11](#_Toc192022984)

[Sequences in *ontocomc* 13](#_Toc192022985)

[*ontocomc* object properties 13](#_Toc192022986)

[*ontocomc* data properties 15](#_Toc192022987)

[*ontocomc* and SWRL Rules 15](#_Toc192022988)

[*ontocomc* and *Owlready2* 17](#_Toc192022989)

[Evaluation 20](#_Toc192022990)

[Application to the Bach chorale corpus 21](#_Toc192022991)

[Future directions 27](#_Toc192022992)

[Appendix I: A bird’s-eye view of tonal functional harmony 28](#_Toc192022993)

[Appendix II. Chord taxonomy in *CofMC* 31](#_Toc192022994)

[Appendix III: Methods extending *ontocomc* classes 32](#_Toc192022995)

[Appendix IV: The Tristan motive, chord analysis 35](#_Toc192022996)

[Appendix V: The piano range in MIDI pitch numbers 37](#_Toc192022997)

[Appendix VI: Herzliebster Jesu (phrase 1), chord analysis 38](#_Toc192022998)

[References 41](#_Toc192022999)

# Introduction

Chord recognition (Harte etc.), annotation,

Ontologies for chord recognition

chord feature extraction

# Related work

Harmonic tension

Chord representation (Cambouropoulos, unified etc.)

Ontologies for harmony

synthetic chorales

# Hindemith’s *Craft of Musical Composition*

*“Our old friend Harmony, once esteemed the indispensable and unsurpassable teaching method, has had to step down from the pedestal upon which general respect had placed her”* (1)

In 1937, Paul Hindemith[[1]](#footnote-2) published the first book (Theoretical Part) of his *Craft of Musical Composition* (2). Two further volumes were written in the 40’s, dealing with the practical aspect of writing in 2 and 3 parts. A fourth book on four-part writing was to conclude the series but was left incomplete at the time of his death.

A black and white image of a musical note

AI-generated content may be incorrect.

Figure : Series 1 and 2

Hindemith starts developing his theory by analyzing the nature of the twelve pitch-classes, by means of the overtone series. Notes of the twelve-tone scale arise one by one through quite intricate and lengthy elaborations on the partials produced by a fundamental or *generator* *tone*. The order in which these are produced reflects their relative value in relationship to the fundamental. The first ones, namely those one fifth or one fourth away from the generator, bear a close relationship to it; as we progress to further ones in the sequence the relationship grows weaker, until it becomes barely perceptible in the tone lying at the distance of a tritone. In this way, Hindemith constructs his *Series 1*, which ranks all 12 pitch-classes in order of their closeness to the generator tone (cf. Figure 1). With this series our author does radically away with the concept of keys and tonality.

Intervals enter next: Hindemith examines the simultaneous sounding of two tones, this time focusing on the phenomenon of combination tones[[2]](#footnote-3). As a result, he ranks all 12 possible intervals in another series (this is his *Series 2*): in this sequence, one moves from the (harmonically) steadier intervals of the fifth and the fourth to the unstable seconds and sevenths, by way of thirds and sixths. Intervals come in pairs, inverted ones immediately following and therefore slightly inferior to their counterparts. We now come across a concept of primal importance in the theory of CofMC: of the two tones, the one best related to the combination tones (produced by the simultaneous sounding) naturally stands out; Hindemith considers this as the fundamental and calls it the *interval* *root*. This is a wholly original concept and does not correspond to any notion in standard tonal theory. In each pair of intervals in *Series 2*, there is one with the root at its low note; its inverted counterpart has the same root, albeit located high, a fact which accounts for its relative instability (Figure 2). *Series 1* and *Series 2* are Hindemith’s basic building tools, the former providing the large connecting elements, the latter supplying the small building material. (3) The two series may look almost similar at first sight, yet they embody two wholly different concepts. *Series 1* assigns tonal positions to all 12 tones in relation to a given generator (the tonal center, pitch-class C assumed in Figure 1). *Series 2* ranks vertical intervals in diminishing harmonic strength while traversing the series from left to right; the melodic nature of intervals is determined by the same series, only this time in reversed direction (Figure 2). Intervals bear an inherent amount of tension, in a sense a ‘load’, non-decreasing as we move to the right. The octave and the unison sound perfectly transparent, their combination tones coinciding with the two real sounds. The tritone stands at the end: its potential is such that it must necessarily resolve to some other interval. The tone closest to the root of the resolution acts as the *root* *representative* of the tritone, while the other tone features as the *guide-tone*. (Figure 3)

A drawing of a musical instrument

AI-generated content may be incorrect.

Figure : Interval roots in Series 2

Hindemith is a harsh critic of the traditional (tonal) harmonic theory. In his view, tertian harmony offers a much too restrictive view of the wealth and complexity of tone simultaneities, as they occur in contemporary music. Indeed, for him a chord is any simultaneity whatsoever, whether tertian or other. In this way he dispenses with the concepts of consonance and dissonance altogether. In general, the dynamics between intra-chord[[3]](#footnote-4) intervals determine the very nature of the chord. Interval roots fight to exert their influence on other intervals; the root corresponding to the best (as measured by Series 2) intra-chord interval becomes the *root* (or *degree*) of the chord (Figure 4)

A black and white image of a musical note

AI-generated content may be incorrect.

Figure : Tritones and their root representatives

A close-up of a music note

AI-generated content may be incorrect.

Figure : Chords and their roots

Inversions in the tonal sense do not exist in this system[[4]](#footnote-5). For Hindemith, an inversion comes about not by shifting around the tones of a chord but by transplanting the root upward (3). In fact, chords with the root at the bass tend to be more stable than formations consisting of the same tones but with their root at a higher voice and hence are somewhat better placed in the chord taxonomy.

We have now hit at the core of the CofMC’s harmonic theory: Hindemith assigns all tone simultaneities, irrespective of their number of distinct tones or their complexity, to specific classes and their subclasses. Chords basically classify into two large groups, according to the existence (or absence) of the tritone among their intra-chord intervals. Chords without tritones belong to *Group A*, while chords containing at least one tritone belong to *Group B*. Chords in the latter group lack the stability of those in Group A, precisely because of the tritone which gravitates towards its resolution, entraining the whole chord along. In general, chords are classified in subgroups according to the nature of their intra-chord intervals. The two main groups are further divided, Group A (Group B) into Subgroups I, III and V (Subgroups II, IV and VI). Further subdivisions pertain to chords in root position or inversion, subgroups indexed with 1 or 2 respectively. In this way, Hindemith creates a complete hierarchy of tone simultaneities, subgroups at the lowest level partitioning the space into pairwise disjoint classes (cf. Appendix I)

*Subgroup I* consists of major and minor triads, either in root position (Subgroup I1) or in inversion (Subgroup I2).

*Subgroup II* consists of chords where the tritone is dominated by some stronger interval. In particular, Subgroup IIa contains the dominant (minor) seventh chords in root position; such chords have no major seconds. At least one major second places a chord in Subgroup IIb, further divided into Subgroups IIb1, IIb2 and IIb3. The former two correspond to chords with one tritone, in root-position and inversion respectively, the latter consists of chords with at least two tritones. Incidentally, Subgroup IIb2 contains our familiar inversions of dominant chords and similar constructions.

*Subgroup III*, a ‘rough and unpolished race’ in Hindemith’s colorful description, corresponds to a very large family, in essence any tone simultaneity comprising seconds or sevenths or both (and obviously no tritone). A small subset of those may be identified as the (familiar from tonal harmony) secondary triads with added 7th and their inversions. Again, chords in root position fall under subgroup III1, inversions under III2.

*Subgroup IV*, a ‘strange set of piquant, coarse and highly colored chords’, contains tone simultaneities with minor seconds or major sevenths or both; tritones are again subordinate to stronger intervals. This subgroup is also further divided into IV1 and IV2.

The last two subgroups are rather sparsely populated. They consist of simultaneities with no definite direction, open to more than one interpretation. As a result, no clear root is assigned to them a priori, though this may be remedied at a second stage according to harmonic context; in other words, the same chord may be provided with a different root, depending on the chords surrounding it. *Subgroup V* consists of Group-A chords comprising solely major thirds and minor sixths, while *Subgroup VI* contains Group-B chords with no other intervals but tritones, minor thirds and major sixths[[5]](#footnote-6). These Group-B chords possess no intra-chord interval strong enough to dominate the tritone, which imbues the whole chord with its inherent vagueness.

In Group-B chords, the tone which belongs to some tritone and at the same time forms the strongest (by *Series 2*) intra-chord interval with the root assumes the role of *guide-tone*[[6]](#footnote-7), as in Figure 5. As already explained, chords in Subgroups V and VI do not have a degree but may be assigned one, namely that tone which stands at best distance from the root of the chord before or after.

A black and white image of a musical note

AI-generated content may be incorrect.

Figure : Guide-tones in Group-B chords

Hindemith observes that the traditional (tonal) harmony is contained in Subgroups I, II, V and VI, except for a few seventh and ninth chords in the other subgroups. The vast space of subgroups III and IV lies beyond the boundaries of tonal harmony. In his view, the ‘chromatic’ harmony of CofMC based on the 12-tone scale represents therefore a vital extension of traditional ‘major-minor’ harmonic theory.

With reference to the table in Appendix I, we observe that every move downwards and to the right induces a decrease in tonal value, while a step in the opposite directions causes an increase (2). On the other hand, moving from higher to lower values increases harmonic tension, while a move from lower to higher values brings about the opposite effect. Hindemith denotes these variations in harmonic tension as *harmonic fluctuation*. Variations may be gradual or sudden according to the relative values of successive chords and is visualized with subgroup numerals and a sort of harmonic crescendo and decrescendo, as in Figure 6.

A black and white image of a music note

AI-generated content may be incorrect.

Figure : The interplay between degree progression, the two-voice framework and harmonic fluctuation

In every chordal progression, the two leading lines are the bass and the next most important above it[[7]](#footnote-8). Their combination is called the *two-voice framework*. These two voices determine the general direction of harmonic activity. Intermediate lines are not relevant at this level; their purpose is to supply each chord with the intra-chord intervals that will determine its place in the hierarchy and regulate harmonic fluctuation from one chord to the next. The *degree* *progression* is another vital feature, in some sense a codification of the whole sequence[[8]](#footnote-9). Indeed, a bad degree sequence is usually evidence of a faulty chord progression (4). As was to be expected, degree intervallic movements are governed by *Series 2*.

Figure 6 demonstrates the three forces at play. It is noteworthy that the degree progression consists in this case of a single pitch-class (C). The two-voice framework exhibits a peak at the fourth chord (minor seventh), a release at the following chord and a higher peak at the penultimate one (tritone). The harmonic fluctuation follows a smoother line, keeping a steady tension for most of the progression before stepping up to its highest point at the penultimate chord. As can be seen, there is constant interaction between these forces, working in the same direction or balancing themselves out in various ways.

The degree progression is most important in setting up the ‘key’ of a piece of music. The term *key* is obviously not used as in the major/minor schema. It rather refers to the establishment of a fundamental tone and the arrangement of other tones in the sequence according to their place in *Series 1*. Naturally, the repeated appearance of some tone tends to increase its relative importance. In addition, the fundamental tone (the tonic center, in this case) will be chosen so that its closest relatives are present and prominent in the sequence. A loss of balance in this sense may bring about the gradual or abrupt rise of a new fundamental tone and impose a modulation. In practice, the degree progression can be used along with (or sometimes against) the harmonic fluctuation in order to provide variety and purpose to the music texture. For instance, a degree progression may be blurred by superposing complex chords from Subgroups III and IV, in which case degree progression and fluctuation will be in conflict (3). Local tonic centers may form another sequence at the next higher level: a fundamental tone will emerge in the same way, this time indicating the tonic center of a more extended excerpt, related to the tonic centers of its various subsections.

In broad terms, non-chord tones in CofMC are treated in a similar way to that of tonal harmony. Although the occurrence of any tone on the music surface establishes potentially a new simultaneity, Hindemith only tags the latter as an independent chord provided that the tempo is slow enough. Otherwise, the transitional simultaneities produced are to be considered as “chord splinters or off-shoots” (2), thus sharing the features (e.g. the degree, guide-tone, subgroup etc.) of the main chords they are appended to.

There is no doubt that Hindemith has erected an original and, in many respects, even ground-breaking harmonic theory. Naturally, his unyielding rejection of tonal harmony does sound questionable when analyzing music obviously based on tonal precepts, despite his efforts to anticipate and nullify critics: “… to musicians irretrievably engrossed in conservatism, not only will methods of composition following our ideas appear fantastic and unartistic; it will even be denied that they are workable” (5). On the other hand, this theory is not restricted to some particular idiom; instead, it aspires to encompass a wide range of genres from medieval to modern music, attempting to provide universal rational principles for solid music making. Particularly noteworthy is his quantization of harmonic tension through chord taxonomy, an area virtually unexplored within the tonal framework. At any rate and leaving music-theoretical reservations behind, in the following we shall mainly focus on the ontological aspect of his theory of chords.

# The *ontocomc* ontology

Short intro to ontologies, OWL and Protégé

Hindemith’s theories in CofMC supply a natural springboard to develop a knowledge representation scheme on tone simultaneities. Not only does his exhaustive chord taxonomy lend itself quite naturally to an ontological re-interpretation; interval roots, chord degrees, guide-tones as well as the specific subgroup a chord belongs to represent a powerful set of features describing a chordal formation. Indeed, the fact that any simultaneity whatsoever, not only the tertian formations of tonal harmony, has the same right to such a characterization is one of the most powerful levers in this scheme. An ontology based on these principles therefore provides us with the tools to perform automatic chord annotation (in CofMC’s terms, of course) of all tone simultaneities occurring in a musical piece and to assess the level of harmonic tension they induce.

Our ontology is called *ontocomc* and was designed using Protégé 5.5.0. In essence, *ontocomc* is anchored on four primary concepts, represented by OWL classes *:PitchClass, :Interval, :Chord* and *:Sequence.*

## Class *:PitchClass*

In the equal temperament tuning system of western music, the octave is divided into 12 equal parts, reflected in our 12 known pitch classes; these are modeled via class *:PitchClass* and its subclasses, as in Figure 8. Each of these subclasses is populated with an individual of identical spelling (albeit in lowercase), as well as all its enharmonically equivalent spellings used in common music practice; individuals *:a, :gss* (for G##)*, :ghh[[9]](#footnote-10)* and *:bbb* (for Bbb)are all declared at the outset as members of the same class *:A* (actually, they are also asserted as identical individuals[[10]](#footnote-11)). There is one more pitch class, *:PitchClassUndef*, to denote tones whose pitch is unknown or unspecified. Hindemith maintains that the nature of a chord and its individual features are basically invariant to translations of its constituent tones across octaves[[11]](#footnote-12), obviously as long as their order is not inverted, therefore the central concept in *:ontocomc* is that of the pitch-class and not the pitch.

## Class *:Interval* and taxonomy

*ontocomc* classifies intervals (except tritones) into disjoint concepts *:RootLow* and *:RootHigh* (and their subclasses, as in Figure 9). Together, these two classes form superclass *:NormIntv*, which is itself disjoint to *:Tritone.* Finally, *:UndefinedInterval* contains intervals with at least one note declared as *:PitchClassUndef*. Subclasses *:NormIntv, :Tritone* and *:UndefinedInterval* constitute a partition of class *:Interval.*

One of the basic ideas in *ontocomc* lies in the explicit definition of each *:Interval* sub-concept; in Figure 9, for instance, class *:Maj2nd* is defined by enumerating all 12 possible combinations of two pitch-classes lying two semitones apart. Class *:UndefinedInterval* on the other hand is defined as *(high some :PitchClassUndef) or (low some :PitchClassUndef)*

A screenshot of a computer

AI-generated content may be incorrect.

Figure : The PitchClass concept and its subclasses

## Class :C*hord* and taxonomy

The *ontocomc* chord taxonomy keeps close track of Hindemith’s specifications in CofMC. Figure 10 shows the *:Chord* hierarchy and its description in Protégé. At the first level, there is the basic partition into *:GroupA* and *:GroupB*.

*:SubgroupV* of *:GroupA* models augmented triads by requiring that all intra-chord intervals belong to the class union *(:Maj3rd or :Min6th or :UndefinedInterval). :SubgroupIII* refers to *:GroupA* chords with at least one second or seventh and is thus defined as *:GroupA and (:hasInterval some (:Maj2nd or :Maj7th or :Min2nd or :Min7th)).* Finally, since pairwise disjoint subclasses *:SubgroupI*, *:SubgroupIII* and *:SubgroupV* are meant to form a closure axiom on *:GroupA*, *:SubgroupI* is simply defined as *:GroupA and (not (:SubgroupIII or :SubgroupV)).*

*:SubgroupVI* requires of intra-chord intervals to be members of class union *(:Min3rd or :Maj6th or :Tritone or :Octave)*. *:SubgroupIV* is defined as *:GroupB and (:hasInterval some (:Maj7th or :Min2nd)).* Οwing to a closure axiom on *:GroupB* similar to the one on *:GroupA* above, *:SubgroupII* is defined as *:GroupB and (not (:SubgroupIV or :SubgroupVI))*. *:SubgroupII* is further partitioned into *:SubgroupIIa* and *:SubgroupIIb*. The former is defined as class intersection *:SubgroupII and (:hasInterval max 0 :Maj2nd) and (:hasInterval max 1 :Tritone) and :RootPositionChord.* Remaining chords get classified in *:SubgroupIIb*.

Finally, *:ontocomc* provides for an alternative classification of a chord according to whether its root coincides with the bass (*:RootPositionChord*) or lies in some higher voice (*:InversionChord*)[[12]](#footnote-13). This fact is inferred through SWRL rules comparing the chord root with the filler of object property *:note1* (cf. *ontocomc* and SWRL Rules).

A screenshot of a computer

AI-generated content may be incorrect.

Figure : The Interval taxonomy and the definition of major second in Protégé. Hindemith makes absolutely no distinction between enharmonic tones, which he merely considers as alternative spellings: in his view, diminished 4th A–Db sounds the same and is therefore treated no differently than major 3rd A–C#. In this sense, the above taxonomy is exhaustive.

A screenshot of a computer

AI-generated content may be incorrect.

Figure : The Chord taxonomy and its description in Protégé

## Sequences in *ontocomc*

In principle, OWL has no inbuilt support for ordering; basic constructs from the underlying RDF vocabulary, such as rdf:List and rdf:nil, are unavailable in OWL-DL. One may of course model sequences using the pattern for linked lists, where each cell contains some content (the ‘head’) and a pointer to the next cell (the ‘tail’). Although quite familiar to software engineers, this structure may sound questionable from the conceptual point of view, as it makes no clear ontological distinction between the sequence itself and its members. We have used an alternative pattern instead, as introduced in (6): a sequence is here represented directly as an independent entity, and its members are linked to it as constituent parts, in a manner shown in Figure 10.

A chord sequence is modeled as a member of class *:ChordSequence*. In essence, this individual provides a handle to the sequence by linking to the initial chord via object property *:firstChord.* Each and every chord is related to this individual via object property *:constituentChord*. Each chord sequence generates in turn its corresponding sequences of chord degrees, guide-tones and the two-voice framework.

A diagram of protein

Description automatically generated

Figure : The Sequence model used in ontocomc (courtesy of (6))

## *ontocomc* object properties

The role hierarchy in ontocomc is displayed in Figure 11. In the following, the *[Domain >> Range]* syntax is used as a shorthand to denote the domain and range of a property.

* *:l*ow and *:high [:Interval >> :PitchClass]* denote the lower and upper voice of an interval. Both properties are functional.
* *:intervalRoot [:Interval >> :PitchClass]* denotes the root of aninterval
* *:hasNote [:Chord >> :PitchClass]* and functional sub-roles *:note1*, *:note2*, *:note3* and *:note4* denote the constituent pitch-classes of a chord
* *:hasInterval [:Chord >> :ChordInterval]* and functional sub-properties *:intervalXY* (for X,Y=1,2,3,4 and X<Y) denote the six intra-chord intervals
* *:sequentChord [:Chord >> :Chord]* connects a chord to all subsequent ones in the sequence. Each chord in a sequence gets connected to its next one via functional sub-role *:nextChord*; the last chord is typically linked (grounded) to *:chordNil*. Role chain axiom *:sequentChord o nextChord SubPropertyOf :sequentChord* enables reaching out iteratively to all subsequent chords in the sequence (cf. Figure 11). Object properties *:sequentDegree, :sequentGuideTone* and *:sequentFrame* follow the same rationale[[13]](#footnote-14).
* *:constituentChord [:ChordSequence >> :Chord]* connects a chord sequence to its members. Functional subrole *:firstChord* links the sequence to the first chord in line, as already explained. Role chain axiom *:constituentChord o :nextChord SubPropertyOf :constituentChord* extends linking to each and every chord in the sequence. Degree sequences, guide-tone sequences and the corresponding two-voice framework are modeled in a similar way.

A screenshot of a computer

AI-generated content may be incorrect.

Figure : The role hierarchy in ontocomc

* Object properties *:degree*, *:frame*, *:guideTone*, *:nextDegree*, *:nextFrame*, *:nextGuideTone, :salientDissonance* and *:salientDissonanceResolution* are filled with chord features inferred upon chord analysis (cf. page 20). All properties have *:Chord* as domain. The first three are functional. The next three are described as super-roles, for instance we have the axiom *nextChord o degree SubPropertyOf nextDegree* and similar chain axioms for *:nextFrame* and *:nextGuideTone.* Functional role *:salientDissonance* denotes the most prominent dissonance (if any) among intra-chord intervals[[14]](#footnote-15); *:salientDissonanceResolution* naturally refers to its resolution in the following chord.

## *ontocomc* data properties

Data properties?? Series2, tied, fermata, semiquaver ornaments, on and off-beat chords (maybe just through odd and even quaver numbers?)

## *ontocomc* and SWRL Rules

Expressivity with description logics and OWL has quite a few limitations. In essence, description logics are a subset of first-order logic, mainly by way of restricting FOL to the use of unary and binary predicates. In this sense, a DL subsumption axiom in the form <class1 *subsumed by* class2> may easily be shown to correspond to the rule class1(?x) -> class2(?x); in this case, the DL axiom acts like syntactic sugar with the advantage of suppressing the use of free variables altogether. Nevertheless, in case an axiom involves more than one distinct free variable, this direct correspondence may cease to exist, and a syntax in some FOL-type language (such as *datalog*) may be required. The combination of *datalog* and DLs gives rise to rule language SWRL[[15]](#footnote-16) (citation required).

DL rules, DL-safety, undecidability etc.

*ontocomc* is supplied with the following set of SWRL rules, as shown in Figure 12:

* two rules for each of the six *:hasInterval* object properties, identifying the high and low notes for some intra-chord interval and adding the corresponding object properties as new facts in the ontology, for instance

*interval12(?crd, ?intv), note2(?crd, ?note) -> high(?intv, ?note)*

This set of rules completely defines intra-chord intervals by denoting both their tones.

* Two rules to attribute an interval root to *:RootHigh* and *:RootLow* intervals and provide the interval with respective object property assertions:

*RootHigh(?int), high(?int, ?note) -> intervalRoot(?int, ?note)*

*RootLow(?int), low(?int, ?note) -> intervalRoot(?int, ?note)*

These two rules identify the root for any *:NormIntv* interval. Tritones are not dealt with here as they cannot be attributed with any root at this stage.

* A further set of rules enables classification in Subgroups I1 or I2, III1 or III2 etc., depending on the degree position in the chord. For instance, SWRL rule

*Chord(?ch), chordHasDegree(?ch, ?n1), note1(?ch, ?n2), SameAs (?n1, ?n2) -> RootPositionChord(?ch)*

effectively ‘pushes’ a chord already in Subgroup III further in the taxonomy and into Subgroup III1, since the latter subgroupis precisely defined as class intersection (:*SubgroupIII* *and* *:RootPositionChord).*

Combining the above DL axioms and SWRL rules enables yielding most of the assertions pertaining to chord features. Nevertheless, assertions about the degree, guide-tone and salient dissonance of a chord cannot easily be inferred from within this framework; to overcome this obstacle, one needs to reach out to Olwready2[[16]](#footnote-17).

A screenshot of a computer program

Description automatically generated

Figure : The set of SWRL rules in ontocomc

6 named chord intervals => dl-safe rules, asserted individuals only

the combination of OWL DL and SWRL is undecidable… … …

## *ontocomc* and *Owlready2*

*Owlready2* is a Python module that allows access to OWL ontologies. It enables ‘ontology-oriented programming’, that is object-oriented programming in which objects and classes are the entities of an ontology. While retaining the expressiveness of formal ontologies, Owlready provides the agility of an object-oriented programming language like Python, permitting the execution of ‘imperative’ lines of code, which is not possible with an ontology or a database alone. (7)

Certain computations in CofMC cannot be performed easily via OWL axioms in *ontocomc*. In particular, the degree of a chord cannot be inferred implicitly by the ontology alone; to identify it, one must compute the ‘best’ (i.e. the one with the *lowest* Series-2 value) among intra-chord intervals. Similarly, the guide-tone of a Group-B chord is selected among tritone tones according to their distance from the chord degree (i.e. finding the minimum Series-2 value). The salient dissonance of a chord is the intra-chord interval with the *highest* Series-2 value (>=7). Such comparisons and value ordering may represent a challenge to OWL expressivity. This may be circumvented by loading *ontocomc* in *Owlready2*: the computation of degrees, guide-tones and salient dissonances can be implemented in a Python environment, respective assertions inferred and then loaded in the ontology.

Owlready2 permits extending *ontocomc* classes with additional methods (cf. Appendix). In particular, class *:Interval* is extended with methods

* *root(self)* returns the low (high) tone of the interval, depending on whether the interval is classified as *:RootLow* (*:RootHigh*).
* *series\_1(self)* returns the Series-1 value of the high note of an interval in relation to a generator tone coinciding with the low note.
* *series\_2(self)* returns the Series-2 value of the interval.

The following methods extend class *:Chord*

* *notes()* displays the four notes of the chord as items in a Python list. For instance, a C major triad may produce the following list:

[ontocomc.c, ontocomc.g, ontocomc.pitchClassUndef, ontocomc.e][[17]](#footnote-18)

* *intrachord()* collects information on intra-chord intervals of a chord, notably their names and Series-2 values.
* *find\_degree()* singles out the ‘best’ among intra-chord intervals by comparing their Series-2 values and thus identifies the chord degree, at the same time asserting the fact in the ontology.
* *find\_guidetone()* checks all intervals between tritone tones and the degree of a Group-B chord, singles out the ‘best’ among them and thus identifies the guide-tone of the chord, at the same time asserting the fact in the ontology. For Group-A chords, the guide-tone is by convention set to coincide with the chord degree.
* *salient\_dissonance()* returns the high and low note positions of the most *dissonant* (if any) intra-chord interval, i.e. some minor or major second, minor or major seventh or tritone with the highest Series-2 value.
* *derove()* attempts to assign a degree to a chord belonging to Subgroups V or VI[[18]](#footnote-19) depending on the neighboring chords. The method first checks for the degree (if available) of the preceding chord; failing this, it checks for the root of the succeeding chord; failing again, the chord in question is left in roving state.

Methods extending class *:Sequence* such as *seq\_features(), seq\_derove()* carry over the above rationale to a whole sequence of chords. The former method repeatedly applies *find\_degree()*, *find\_guidetone()* and *salient\_dissonance()* asserting respective chord features for the whole sequence. During a second pass, the sequence is checked for roving chords in order to apply *seq\_derove()*.

Method *seq\_analysis()* combines the above two methods. In essence, it implements one of the core tasks of *ontocomc* ontology, namely chord annotation for each chord in the sequence. Figure 13 demonstrates the procedure with the help of a toy 5-chord sequence: an initial C-major chord moves to a D-minor one by way of two distinct diminished chords, the former repeated twice in a different position. The sequence and its constituent chords are first asserted from within a Python/Owready script before method *seq\_analysis()* is applied. The reasoner needs to be called twice during execution: once to classify intra-chord intervals (during which chords also get automatically classified to respective subgroups) and once again during the de-roving process, to classify newly created intervals between roving chord tones and their neighboring degrees. Both steps are necessary, since fresh individuals always need to get classified in order to enable further computations; in both cases, Olwready2 first creates all intervals for the whole sequence; the HermiT reasoner is then called once to classify them all in one batch, so as to minimize the time burden. These two reasoner calls account for almost the entire runtime (about a minute, in this case). Two chords have been ‘de-roved’ in our example, namely the ones following C-major and preceding D-minor. The diminished seventh chord in the middle, flanked on both sides by (originally) roving chords, has not been affected. The output displays basic chord features for all chords: the Subgroup each chord belongs to, computed degrees and guide-tones together with their respective positions in the chord, salient dissonances (where applicable) and their resolutions together with interval types.

|  |
| --- |
| *cmaj = hind.Chord(name='cmaj',note1=hind.c,note2=hind.e,note3=hind.g,note4=hind.c)*  *dim1 = hind.Chord(name='dim1',note1=hind.d,note2=hind.f,note3=hind.ab,note4=hind.b)*  *dim2 = hind.Chord(name='dim2',note1=hind.f,note2=hind.ab,note3=hind.b,note4=hind.d)*  *dim3 = hind.Chord(name='dim3',note1=hind.e,note2=hind.g,note3=hind.g,note4=hind.ch)*  *dmin = hind.Chord(name='dmin',note1=hind.f,note2=hind.f,note3=hind.a,note4=hind.d)*  *cmaj.nextChord = dim1*  *dim1.nextChord = dim2*  *dim2.nextChord = dim3*  *dim3.nextChord = dmin*  *dmin.nextChord = hind.chordNil*  *hind.toy\_seq = hind.ChordSequence(name='toy\_seq',firstChord=cmaj)*  *hind.toy\_seq.seq\_analysis()*  Hindemith's Craft of Musical Composition  Harmonic analysis of chord sequence 'toy\_seq':  [ontocomc.c, ontocomc.e, ontocomc.g, ontocomc.c]  [ontocomc.d, ontocomc.f, ontocomc.ab, ontocomc.b]  [ontocomc.f, ontocomc.ab, ontocomc.b, ontocomc.d]  [ontocomc.e, ontocomc.g, ontocomc.g, ontocomc.ch]  [ontocomc.f, ontocomc.f, ontocomc.a, ontocomc.d]  running the reasoner to classify intra-chord intervals for all chords...  running the reasoner to classify intervals between roving chord notes and the neighbor degree...  deroved chord 'ontocomc.dim1', degree is 'ontocomc.f'  deroved chord 'ontocomc.dim3', degree is 'ontocomc.g'  cmaj: [ontocomc.SubgroupI1] (1, ontocomc.c) (1, ontocomc.c)  dim1: [ontocomc.SubgroupVI, ontocomc.InversionChord] (2, ontocomc.f) (0, ontocomc.pitchClassUndef)  salient dissonance: ontocomc.dim1\_24, [ontocomc.Tritone, ontocomc.ChordInterval24] >>> [ontocomc.Tritone, ontocomc.ChordInterval24]  dim2: [ontocomc.SubgroupVI, ontocomc.InversionChord] (0, ontocomc.pitchClassUndef) (0, ontocomc.pitchClassUndef)  salient dissonance: ontocomc.dim2\_24, [ontocomc.Tritone, ontocomc.ChordInterval24] >>> [ontocomc.Tritone, ontocomc.ChordInterval24]  dim3: [ontocomc.SubgroupVI, ontocomc.InversionChord] (2, ontocomc.g) (0, ontocomc.pitchClassUndef)  salient dissonance: ontocomc.dim3\_34, [ontocomc.Tritone, ontocomc.ChordInterval34] >>> [ontocomc.Perf4th, ontocomc.ChordInterval34]  dmin: [ontocomc.SubgroupI2] (4, ontocomc.d) (4, ontocomc.d) |

Figure : Chord analysis, example

At this point, the ontology has been duly updated with facts about degrees, guide-tones and salient dissonances of the chords and may be saved in an .owl file. All remaining chord features (frame, next degree, next guide-tone, next frame and salient dissonance resolution) as well as the specific position of each chord in the chord taxonomy have also been inferred during the process.

A further example automatically annotating chords of the opening motive from Wagner’s *Tristan* may be observed in the Appendix.

# Evaluation

Bla bla bla Questions the ontology answers to:

**Given an interval of two pitch-classes, we can determine the interval root.** When running the reasoner, the interval will match one of the 12 interval types, i.e. the definition of either class *:Tritone* or one of the subclasses of *:NormIntv* (*:Maj2nd*, *:Min3rd*, *:Perf4th* etc.). The interval will be assigned to the respective subclass and also asserted as a member of either *:RootLow* or *:RootHigh*, unless it is a tritone in which case there is no root.

**Given the 4 pitch-classes of a chord, we can classify the chord in its specific subgroup**. As a preparatory step, we need to provide for six fresh individuals representing the intra-chord intervals. The reasoner will use SWRL rules to confer low and high notes on them from the chord notes set. It will then check intra-chord interval types against subgroup definitions and descriptions to push the chord down the chord taxonomy. It must be noted that the last level of the taxonomy, where subgroups are subscripted with 1 or 2 according to the position of the chord degree, can only be inferred provided the chord root has been asserted in the ontology.

**We can perform automated feature extraction for all chords in a chord sequence.** This sequence may typically be a Bach chorale and chord features include the subgroup, the root and guide-tone, the salient dissonance, their resolution in the next chord, the chords following in the sequence etc.

**Given a chord sequence, we can draw up the corresponding degree sequence, guide-tone sequence and the two-voice framework.** In broad terms, Hindemith considers the design of a new chord progression in two layers. The first of them sets up the basic construction by using these sequences as building blocks. Middle voices are then added by way of voice-leading so as to produce the intended harmonic fluctuation.

**Having stored chords and their features in the ontology, specific chordal patterns may emerge.** We can put forward queries about cadences and patterns in general (not only over consecutive chords), propose alternative chords sharing specific features with their counterparts and/or satisfying additional constraints (e.g. their highest note coinciding with the melody, suitably blending together with sequent chords etc.)

… … … … … …

… … … … … …

# Application to the Bach chorale corpus

A chorale is a hymn of the Lutheran church. The melody and words are intended to be sung by the whole congregation and many of them date from as early as the 16th century, indeed several are attributed to Martin Luther himself. It soon become customary to invest the melody with a full harmonic accompaniment for organ (or sometimes for a 4-part choir). This practice reached its peak with late-baroque masters of the 18th century, not least Johann Sebastian Bach, who composed more than 200 chorales as choral parts of his cantatas or his major choral works.[[19]](#footnote-20) In these pieces, the melody is usually entrusted to the highest voice and combined with counterpoint in 3 additional lines, the latter supplying harmonic purpose and direction. In general, Bach chorales are considered as perfect artifacts imbued with a deep and thorough sense of the dynamics of tonality. Although sometimes carried out in a stern, mainly homophonic style, they are most often treated in an intricate contrapuntal style which aims at providing the musical surface with harmonic variety and richness. The lower voices are moving as independent entities while at the same time contributing to the harmonic construction of the whole and allowing for seamless transitions between tonal regions. In his chorale writing, Bach achieves a perfect balance of the horizontal and the vertical element through extensive and deft use of non-chord (nonessential) tones investing the bass and intermediate lines with a melodic content of their own, quite often at odds with static harmonies but without ever compromising tonal unity. Tonicizations and modulations are ingeniously thought out, never as a display of virtuosity but as an expressive comment on the melody or the words of the hymn (8).

Bach chorales are collected and published in (9). As source material for this section we have used *.midi* music files of the entire Bach chorale corpus posted at <https://kunstderfuge.com>. In broad terms, a MIDI file allows the representation of a piece of music in symbolic form. It contains a list of MIDI messages, the most important of which are the note-on and the note-off commands, which correspond to the start and the end of a note respectively. Each note-on and note-off message is, among others, equipped with a MIDI note number, a value for the key velocity, a channel specification, as well as a timestamp. The MIDI note number is an integer between 0 and 127 and encodes a note's pitch from C0 to G♯9 (cf. Appendix VI). Velocity corresponds to the intensity of the note being played, in essence encoding dynamics. A channel indicates the instrument (or voice) that performs the notes in question or display metadata about the piece (see Figure 14)

MIDI files representing Bach chorales are first transformed into CSV files. Figure 15 displays the head of such a file corresponding to the chorale ‘Herzliebster Jesu’ from the St Matthew Passion (BWV 244); channel 2 corresponds to the soprano voice singing the beginning of the chorale melody (B-B-B-A#-F#-B etc.). The first entry in each line corresponds to the timestamp. Dynamics are typically kept at the same level throughout, so a constant velocity appears as the last entry in each note-on line (zero velocity in each note-off line indicates instant release).

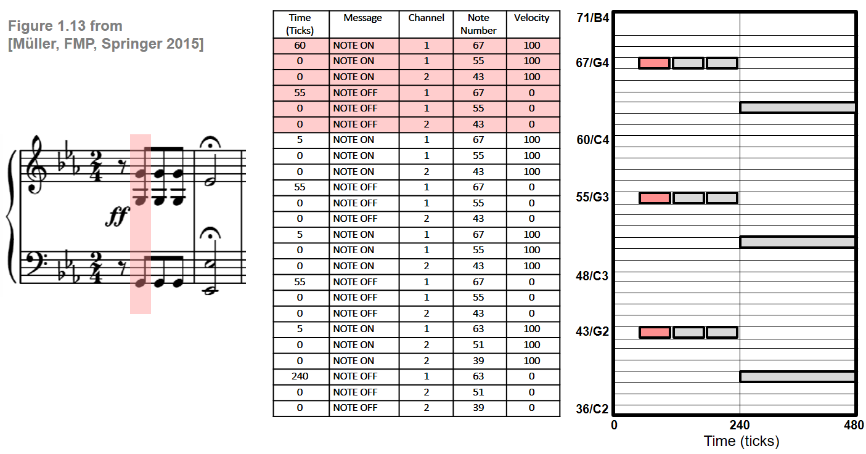


Figure : The four-note opening motive of Beethoven's Fifth. From left to right: a sheet-music representation, a simplified MIDI file and a piano-roll representation (10)

|  |
| --- |
| ['0, 0, Header, 1, 6, 1024\n',  '1, 0, Start\_track\n',  '1, 0, Time\_signature, 4, 2, 24, 8\n',  '1, 0, Key\_signature, 2, "minor"\n',  '1, 0, Tempo, 625000\n',  '1, 49152, End\_track\n',  '2, 0, Start\_track\n',  '2, 0, Title\_t, "Instrument 1"\n',  '2, 3072, Note\_on\_c, 0, 71, 96\n',  '2, 4096, Note\_off\_c, 0, 71, 0\n',  '2, 4096, Note\_on\_c, 0, 71, 96\n',  '2, 5120, Note\_off\_c, 0, 71, 0\n', |

Figure : the first few lines of a CSV file for Bach’s chorale ‘Herzliebster Jesu’

Information from CSV files is then processed in a way to provide the 4 notes of all tone simultaneities formed vertically at quaver time quantization[[20]](#footnote-21), as shown in Figure 16. Chords are entered next as fresh individuals in the ontology and connected to one another to set up the chord sequence. Chord analysis may now be performed and the ensuing ontology, fully updated with chord features, saved in an *.owl* file. The whole process takes place in a *Python / Owlready* environment.

|  |
| --- |
| * transform input MIDI file to CSV file (using py\_midicsv package) * create a Pandas DataFrame from CSV file * drop any irrelevant columns (e.g. velocity etc.), keeping channel, timepoint, on/off indications and MIDI-pitch * focus on channels 2-5 (soprano, contralto, tenor and bass respectively). For each of those channels:   + drop the two first and the very last lines (see Figure 14)   + drop the channel column (irrelevant hereafter)   + assign a quaver number. Keep entries where the timestamp is a multiple of 512, the duration of a quaver in standard MIDI files   + convert MIDI-pitch numbers to equivalent pitch-classes   + assert the note (pitch-class) with Boolean value *‘tied’ = False* (default value)   + assert the note (pitch-class) with value *‘semiqv\_orn’ = ‘None’* (default value, indicates that the note is not part of some semiquaver ornament)   + assert the note (pitch-class) with Boolean value *‘fermata’ = False* (default value, indicates that the note is not part of a fermata)   + for quaver numbers corresponding to the space between successive note-on and note-off signals (i.e. while a note has been issued and is still sounding), pad with the pitch-class issued at the note-on sign. Note will be tied to the previous one   + for quaver numbers corresponding to eventual pauses between a note-off sign and the next note-on one, fill in with *ontocomc.PitchClassUndef*   + set *quaver* as the DataFrame index |

Figure 16: Extracting chord notes from MIDI files (pseudocode)

A sheet music with notes

AI-generated content may be incorrect.

Figure : J. S. Bach’s chorale ‘Herzliebster Jesu’ BWV 244.46 (courtesy of www.[bach–chorales.com](https://www.bach-chorales.com/Index.htm) by [Luke Dahn](http://www.lukedahn.net/Index.htm))

|  | **timepoint** | **on/off** | **pitch** | **pitchclass** | **tied** | **semiqv\_orn** | **fermata** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **quaver** |  |  |  |  |  |  |  |
| **7** | 3072 | Note\_on\_c | 71 | ontocomc.b | False | None | False |
| **8** | 0 | Note\_on\_c | 71 | ontocomc.b | True | None | False |
| **9** | 4096 | Note\_on\_c | 71 | ontocomc.b | False | None | False |
| **10** | 0 | Note\_on\_c | 71 | ontocomc.b | True | None | False |
| **11** | 5120 | Note\_on\_c | 71 | ontocomc.b | False | None | False |
| **12** | 0 | Note\_on\_c | 71 | ontocomc.b | True | None | False |
| **13** | 6144 | Note\_on\_c | 70 | ontocomc.bb | False | None | False |
| **14** | 0 | Note\_on\_c | 70 | ontocomc.bb | True | None | False |
| **15** | 7168 | Note\_on\_c | 66 | ontocomc.gb | False | None | False |
| **16** | 0 | Note\_on\_c | 66 | ontocomc.gb | True | None | False |
| **17** | 8192 | Note\_on\_c | 71 | ontocomc.b | False | None | False |
| **18** | 0 | Note\_on\_c | 71 | ontocomc.b | True | None | False |
| **19** | 9216 | Note\_on\_c | 73 | ontocomc.db | False | None | False |
| **20** | 0 | Note\_on\_c | 73 | ontocomc.db | True | None | False |
| **21** | 10240 | Note\_on\_c | 74 | ontocomc.d | False | None | False |
| **22** | 0 | Note\_on\_c | 74 | ontocomc.d | True | None | False |
| **23** | 11264 | Note\_on\_c | 74 | ontocomc.d | False | None | False |
| **24** | 0 | Note\_on\_c | 74 | ontocomc.d | True | None | False |
| **25** | 12288 | Note\_on\_c | 76 | ontocomc.e | False | None | False |
| **26** | 0 | Note\_on\_c | 76 | ontocomc.e | True | None | False |
| **27** | 13312 | Note\_on\_c | 74 | ontocomc.d | False | None | False |
| **28** | 0 | Note\_on\_c | 74 | ontocomc.d | True | None | False |
| **29** | 14336 | Note\_on\_c | 73 | ontocomc.db | False | None | False |
| **30** | 0 | Note\_on\_c | 73 | ontocomc.db | True | None | False |

Figure : Bach’s ‘Herzliebster Jesu’, opening phrase, soprano line

We can monitor the whole process using the opening phrase of J. S. Bach’s chorale ‘Herzliebster Jesu’ from the St-Matthew Passion as in Figure 17. We begin by entering a *\*.mid* file at the input and extracting chord notes and relevant information therefrom, one Pandas Dataframe for each voice (the one corresponding to the soprano line is shown at Figure 18). Chords are entered next as new individuals by filling object properties *:note1, :note2, :note3, :note4* and *:nextChord* with the 4 chord notes and the chord at the next quaver, respectively. Chords are indexed with their quaver number, odd (even) numbers corresponding to (non) accented quavers in the bar. In naming chords, we are following syntax <chord sequence name>\_<quaver number>, therefore the first member of chord sequence *:herzliebster* will be named *:herzliebster\_7* (NB the incomplete first bar at Figure 17). Performing chord analysis on this opening phrase by calling method *seq\_analysis()* yields results as in Appendix, page 36.

|  |
| --- |
| * Isolate the index of a voice DataFrame (cf. Figure 18) indicating the quaver numbers in the sequence. Use it as the x-axis * Match subgroup names from I1 to VI with positive decimal numbers indicating increasing tension levels. * For each chord in the sequence, trace its subgroup in the ontology and register its corresponding tension level in the y-axis * Plot the harmonic fluctuation (tension curve) of the chord sequence |

Figure : Plotting harmonic fluctuation (pseudocode)

Graphically displaying harmonic fluctuation can be implemented by means of the ontology updated will chord features (cf. Figure 19). Charts corresponding to the second phrase of *Herzliebster* *Jesu* are shown in Figure 20. A word of explanation is in order at this point. We have largely been using the term ‘chord analysis’ instead of ‘harmonic analysis’; the distinction here is intentional. A full-fledged harmonic analysis based either on tonal or on Hindemith’s precepts should normally account for non-chord tones in the context of the main chords. One may well use the *ontocomc* ontology in this sense, yielding the chart in Figure 20b[[21]](#footnote-22); harmony obviously moves by crotchets and non-chordal formations have largely been suppressed. As is well documented, isolating the main chords among the set of all tone simultaneities is by no means an evident task, typically entrusted to human experts or automated procedures. Either way, the final choice is inherently equivocal and open to multiple interpretations. For example, Quavers 47 and 48 (the last two quavers of the first line in Figure 17) could jointly be interpreted as either a minor seventh chord on A or a half-diminished chord on G#. Neither of these chords though would be expected to occur as a self-contained entity in Bach’s epoch and personal style. A more plausible interpretation would consider the harmonic pace slowing down to a minim in this case; all intermediate simultaneities would explain themselves by means of an extensive voice-leading over a simple B-minor triad dominating the whole second half of the bar; this more insightful analysis is displayed in Figure 20c.

|  |
| --- |
| A diagram of a graph  AI-generated content may be incorrect. a  A graph with a line  AI-generated content may be incorrect. b  A graph with a line  AI-generated content may be incorrect. c |

Figure : Herzliebster Jesu, phrase 2, harmonic fluctuation

The chart in Figure 20a illustrates a radically different approach. Here, time quantization has been (quite doggedly) kept at a strict quaver, allowing us to register all time simultaneities occurring at this pace. This is admittedly not a ‘proper’ harmonic analysis; nevertheless, it supplies several quite important insights into harmony in general and Bach’s style in particular. Notwithstanding their transitory nature, non-chord simultaneities induce a momentary tension in the music texture, which should be monitored and examined. This tension is comparatively easy to pinpoint on printed paper or perceive during an audition, though elusive to identification and quantification through tonal analytical procedures. Referring once again to our example chorale, the first phrase would largely produce similar charts, except for a passing simultaneity in the third bar; we would expect a generally smooth curve, apart from a couple of diminished chords. The ragged curve of the second phrase comes in sharp contrast and indicates the deliberate injection of passing dissonances and of a longer state of high tension at quavers 46-50[[22]](#footnote-23). Regulating this tension is a distinct style feature and a powerful compositional tool in Bach’s hands. In this case, he obviously decides to elaborate and thicken his harmonic texture as the chorale develops and we can closely follow and analyze this process.

In most cases, non-chord tones form non-tertian chords which will typically be members of subgroups III and IV. In Bach’s style, as we may see in Figure 20a, chords in Subgroup III and IV imply voice-leading by means of non-chord tones and are expected to be preceded and followed by chords in low-numbered subgroups (corresponding to tertian simultaneities) . In general, an off-beat (i.e. at some even quaver number) occurrence of such chords will correspond to passing or neighboring notes (typically in quavers), as at time-points 38, 40, 42 or 44, while an on-beat occurrence will account for suspensions, appoggiaturas and similar patterns, as at time-points 47 and 49. Regarding the latter examples, we may further observe that neither resolves immediately to a chord in Subgroup I or II, a situation highlighting a complex voice-leading which may be further analyzed[[23]](#footnote-24). Outlining the harmonic fluctuation becomes thus a tool to detect and analyze voice-leading as well as simulate the use of non-chord tones when shaping a chord sequence.

# Future directions

* Relations to other ontologies
* Identifying chordal patterns (cadences etc.) to forge Bach’s style / 3-grams (on quavers) differentiating on landing at on- / off-beat

Tonal centers

Statistical analysis of Bach chorale corpus

Use of chord features in deep learning

Acknowledgements. Protégé., …

# Appendix I: A bird’s-eye view of tonal functional harmony

In a nutshell, harmony is the art of combining sounds in simultaneities called *chords*. It deals with the theory (description, identification and relation) of chords and elaborates on constructing and connecting them to one another (10). A basic building block is the *interval* between two tones, in some sense their ‘distance’, as determined by their frequency ratio. The smallest such interval is the second, either a *semitone* or a *whole* *tone*, the latter consisting of two consecutive semitones. In general, intervals are denoted by the number of intermediate tones: the interval A-C, for instance, is a third (counting the 3 first notes in the series A, B, C, D, E, F and G). Thirds, sixths and sevenths are further classified as *major* or *minor*, according to the number of consecutive semitones they contain: back to our example, A-C is a minor third (consisting of 3 semitones in total, whereas a major third would comprise 4). Fourths and fifths are *perfect* (when with 5 and 7 semitones, respectively), *augmented* or *diminished*. Specifically, a diminished fifth is equivalent to an augmented fourth; they comprise 6 semitones (or 3 whole tones) each and are also called *tritones*.

Intervals may be

* Consonant: minor and major thirds, perfect fourths and fifths, minor and major sixths, unisons and octaves.
* Dissonant: practically all the rest, i.e. minor and major seconds, minor and major sevenths and all diminished and augmented intervals without exception[[24]](#footnote-25).

Generally speaking, a piece of music composed in the tonal idiom is based on a tonality, usually a major or minor key, sometimes one of the older church modes (lydian, dorian etc.). Each such key or mode is a distinct sequence of 7 pitch-classes (out of 12 in total) in preset distances. The first note is called the *tonic* or first degree, followed by the second, the third etc., each tagged by a latin numeral (I to VII).

In tonal harmony, chords are formed by stacking notes upon one another, always at the distance of a third, following the *tertian principle*. A three-tone chord or *triad* consists therefore of two successive thirds, the outer tones lying a fifth apart. The lower note (the bass) is called the *root*, the other ones are named the *third* and the *fifth* of the chord.

Tonal chords are classified as

* Consonant, namely major (resp. minor) triads, in which the third lies a major (resp. minor) third above the root and the fifth of the chord always a perfect 5th apart
* Dissonant, i.e. any remaining triads and all chords with 4 or more (distinct) tones. Such simultaneities consist of pitch-classes perceived as somehow incompatible to each other and therefore tend towards some sort of reparation or *resolution* of the dissonance (10).

Stacking a fourth tone on top of a triad creates a seventh between the outer voices[[25]](#footnote-26), producing a 7th-chord (inevitably a dissonant one). In case the root cedes its place at the bass to some other chord tone, the chord is denoted as *inverted*. Inversions are typically used to enrich harmonic texture and allow for a smoother voice leading and an improved bass line. A major breakthrough in harmonic thinking occured in the 18th century with Jean-Philippe Rameau (1683 – 1764), who acknowledged individual qualities in chords built on different degrees. He further asserted that these specific qualities are invariant to chord inversions and depend primarily on the *basse fondamentale* (fundamental bass, i.e. the root) of the chord (11). A second major step took place in mid-19th century with Simon Sechter’s (1788-1867) *Stufentheorie* (German for ‘degree theory’), where chord progressions are invested with specific functions and analyzed in relation to the intervals formed between successive chord roots. The concept of harmonic functions is elaborated further in Hugo Riemann’s (1849-1919) *Funktionstheorie* (‘theory of functions’) , describing two competing and counterbalancing harmonic forces at play: the *subdominant* *SD*, mainly represented by the major chord on the fourth degree and the *dominant* *D*, whose principal agent is the major chord on the fifth degree. The former normally leads to the latter and this in turn to the *tonic T* (the major chord on the first degree): the succesion *IV-V-I* (SD-D-T) constitutes the *perfect* *cadence,* a harmonic schema firmly establishing the tonality. Chords on I, IV and V are called *primary*; *auxiliary* *chords* on the remaining degrees typically perform one of the above functions as substitutes to primary chords.

The seven pitch-classes of a mode constitute the *diatonic* environment. By applying *chromatic* inflections to these 7 tones, one may obtain the remaining 5 pitch-classes and enrich the harmonic texture with altered chordal formations, secondary dominants etc. There is now a much wider choice of chords which may expand tonal functions to whole regions across the musical surface. The general tendency is always from the subdominant to the dominant, although the route may become tortuous, gradually gaining momentum before finally landing on the tonic, in what may still be considered as a more or less extended cadence. This course may also include key changes, either transient (*tonicization*) or more or less persistent (*modulation*). Modulations to closely-related keys (during the exposition section of a piece) or more remote ones (in the development section) give prominence to centrifugal tendencies, at the same time providing structure and variety to the musical surface. Transitions between keys are not always sharp and intermediate chords may be open to multiple tonal meanings, a state that Schoenberg calls *vagrant* or *roving*[[26]](#footnote-27) harmony. (12), (13)

The dynamics of tonal harmony are regulated by two complementary forces. On the one hand, one has the vertical aspect of chords and their functions across keys and tonal regions. On the other hand, there is the horizontal element implemented by the simultaneous voice leading of several contrapuntal lines. This interplay between *harmony* and *counterpoint* gains considerable impetus by the use of *neighboring* and *passing* *tones*, *suspensions*, *appogiaturas* etc. These *non-chord tones* are injected on the musical surface in-between occurences of normal chords and are perceived as circumstancial and irrelevant to the overall harmonic scheme. Such tones cause all sorts of transitional vertical simultaneities not directly supported by tonal harmony[[27]](#footnote-28), since these formations typically do not conform to the tertian principle of chordal construction. In principle, these tone simultaneities possess a dissonant nature and therefore induce some tension in the texture; suspensions and appogiaturas normally occur on accented beats and the ensuing tension is rather high, while passing and neighboring tones usually occur off-beat and have generally a milder effect.

# Appendix II. Chord taxonomy in *CofMC*

A sheet of music with notes

AI-generated content may be incorrect.

# Appendix III: Methods extending *ontocomc* classes

The following methods are cited in pseudocode

**Methods extending *ontocomc:Interval***

**def *root(self)*:**

if *self* is classified as RootLow then return the low note of *self*

if *self* is classified as RootHigh then return the high note of *self*

else return *ontocomc*.*PitchClassUndef*

**def *series\_1(self)*:**

*series1\_intvs* = list of interval types, the low note L always considered as the generator tone and the \

high note traversing CofMC’s Series-1, suitably transposed with L as basis

find the type of *self* then return its rank in *series1\_intvs* (1, 2, …, 12)

if *not found* return 13

**def *series\_2(self)*:**

*series2\_intvs* = list of interval types in order of CofMC’s Series-2

find the type of *self* then return its rank in *series2\_intvs* (1, 2, …, 12)

if *not found* return 13

**Methods extending *ontocomc:Chord***

**def notes(self):**

return [*self.note1*, *self.note2*, *self.note3*, *self.note4*]

**def intrachord(self):**

for i in range(len(*self.notes()*):

for j in range(i+1,len(*self.notes()*):

create new individual *ontocomc.<self>\_(i+1)(j+1)* with corresponding low / high notes \

e.g. for chord named ‘*chord\_x’*, i=0,j=1 create interval named ‘*chord\_x\_12*’

run the reasoner to classify new intra-chord interval individuals

for each intra-chord interval calculate its Series-2 value

assert intervals as fillers of respective ontocomc.intervalXY object properties \

e.g. assert triple <*chord1 ontocomc.interval12 chord1\_12*>

return interval names, Series-2 values ordered by Series-2 values

**def find\_degree(self,intrachord\_dataframe):**

# intrachord\_dataframe is the output of method *self.intrachord*

# returns an integer indicating the position of the degree in the chord

if *self* in Subgroups V or VI:

assert *ontocomc.PitchClassUndef* as filler to object property *self.degree*

return 0

    else:

      drop line/s at top of intrachord\_dataframe with Series-2 value = 1 corresponding to \

unisons / octave doublings

assert root of interval in top line as filler to object property *self.degree*

return position (1, 2, 3 or 4) of chord degree

**def find\_guidetone(self,degree\_position,intrachord\_dataframe):**

# intrachord\_dataframe is the output of method *self.intrachord*

# returns an integer indicating the position of the degree in the chord

if *self* in Subgroups V or VI:

assert *ontocomc.PitchClassUndef* as filler to object property *self.guideTone*

return 0

    else:

      if *self* in Group A:

*self.guideTone* = *self.degree*

        return position of chord degree

      else:

identify all chord notes belonging to some tritone

find the best interval between some tritone note and the chord degree

in best interval found, identify note *x* different from degree

assert *x* as filler to object property *self.guideTone*

return position (1, 2, 3 or 4) of guide-tone

**def salient\_dissonance(self,intrachord\_dataframe):**

# intrachord\_dataframe is the output of method *self.intrachord*

# returns a tuple with the note positions of the most prominent dissonance

restrict intrachord\_dataframe to intervals in Series-2 value range [8,12]

if *self* contains no dissonant intervals: return (0,0)

else:

      extract the index of the last line of intrachord\_dataframe (X,Y)

      assert corresponding intra-chord interval as filler for object property \

*ontocomc.salientDissonance* e.g. for chord ‘*chord1’* \

add triplet <*chord1* *chord1.salientDissonance chord1\_XY*>

return (X,Y)

**def derove (self,neighbor):**

# neighbor may be the chord preceding or following the *roving* harmony

# returns a tuple with the (newly determined) degree and the guide-tone of the chord

if *neighbor*.*degree* == *ontocomc.pitchClassUndef*: return (0,0)

    else:

      create fresh individuals for intervals between *self.notes()* and the neighbor degree

      run the reasoner to classify new intervals and compute their Series-2 values

select the interval among them with min Series-2 value

      identify the high note of chosen interval as the chord degree at position X

assert the new degree as filler to object property *self*.*degree* in *ontocomc*

return (X,0)

**Methods extending *ontocomc:ChordSequence***

**def seq\_features(self):**

# updates the ontology with features, notably chord degrees, guide-tones etc.

# applies Chord methods rationale for all chords in the sequence in one batch

for each chord in the sequence:

create fresh individuals for all intra-chord intervals [cf. Chord method *intrachord()*]

      assert new intervals as fillers to Chord object properties *:interval12*, *:interval13* etc.

construct a sequence (list) with intra-chord info for all chords

run the reasoner to classify new intervals #only once for all intra-chord intervals for all chords

for each *intra\_chord\_info* in the sequence:

apply Chord method *find\_degree()* with parameter *intra\_chord\_info*

apply Chord method *find\_guidetone()* with parameter *intra\_chord\_info* and the chord degree

apply Chord method *salient\_dissonance()* with parameter *intra\_chord\_info*

for each chord in the sequence:

if the chord is not dissonant: positions of the resolution of salient dissonance in the next chord are (0,0)

else:

identify the note positions X,Y of the most prominent dissonance of chord *self*

assert triplet <*self ontocomc:salientDissonanceResolution self.next\_XY*>

return analysis\_results, i.e. degrees, guide-tones and salient dissonances

**def seq\_derove(self,analysis\_results):**

for all chords in the sequence:

if chord already possesses a root or is the first or last chord in the sequence: continue

elif preceding chord has a root: neighbor = preceding chord

elif following chord has a root: neighbor = following chord

else: continue # chord will remain at roving state

create new individuals for intervals formed between roving chord notes and the neighbor degree

run the reasoner to classify all new intervals in one batch for the whole sequence

for each chord in the sequence:

compute Series-2 values for intra-chord intervals and single out the min among them

        assert corresponding note of roving chord as filler to *ontocomc.degree()* object property

**def seq\_analysis(self):**

    apply self.seq\_features() to obtain analysis\_results

    apply self.seq\_derove(analysis\_results)

# Appendix IV: The Tristan motive, chord analysis

Diagram

Description automatically generated

The opening motive from Wagner’s *Tristan und Isolde* is arguably the most famous[[28]](#footnote-29) chord sequence of 19th century western music. Of the four tone simultaneities in total, analysts agree in general that two are to be considered as main chords while the remaining two occur through voice leading: the choice of the two main chords is not at all unanimous, though. For instance, theorists advocate two contradictory points of view for the first chord: one may consider the first simultaneity as an appoggiatura to the second (which is then clearly a *French sixth* chord, altered secondary dominant in A-minor) or else acknowledge the first (and much longer) simultaneity as the main one[[29]](#footnote-30) and consider the short second as passing over to the next bar.

A chord analysis of all four tone simultaneities is here on display.

*tristan1 = hind.Chord(name='tristan1',note1=hind.f,note2=hind.b,note3=hind.dh,note4=hind.gh)*

*tristan2 = hind.Chord(name='tristan2',note1=hind.f,note2=hind.b,note3=hind.dh,note4=hind.a)*

*tristan3 = hind.Chord(name='tristan3',note1=hind.e,note2=hind.gh,note3=hind.d,note4=hind.ah)*

*tristan4 = hind.Chord(name='tristan4',note1=hind.e,note2=hind.gh,note3=hind.d,note4=hind.b)*

*tristan1.nextChord = tristan2*

*tristan2.nextChord = tristan3*

*tristan3.nextChord = tristan4*

*tristan4.nextChord = hind.chordNil*

*hind.tristan = hind.ChordSequence(name='tristan',firstChord=tristan1)*

*hind.tristan.seq\_analysis()*

Hindemith's Craft of Musical Composition

Harmonic analysis of chord sequence 'tristan':

[ontocomc.f, ontocomc.b, ontocomc.dh, ontocomc.gh]

[ontocomc.f, ontocomc.b, ontocomc.dh, ontocomc.a]

[ontocomc.e, ontocomc.gh, ontocomc.d, ontocomc.ah]

[ontocomc.e, ontocomc.gh, ontocomc.d, ontocomc.b]

running the reasoner to classify intra-chord intervals for all chords...

running the reasoner to classify intervals between roving chord notes and the neighbor degree...

tristan1: [ontocomc.InversionChord, ontocomc.SubgroupIIb2] (4, ontocomc.gh) (1, ontocomc.f)

salient dissonance: ontocomc.tristan1\_12, [ontocomc.ChordInterval12, ontocomc.Tritone] >>> [ontocomc.ChordInterval12, ontocomc.Tritone]

tristan2: [ontocomc.SubgroupIIb3, ontocomc.RootPositionChord] (1, ontocomc.f) (4, ontocomc.a)

salient dissonance: ontocomc.tristan2\_34, [ontocomc.ChordInterval34, ontocomc.Tritone] >>> [ontocomc.ChordInterval34, ontocomc.Min6th]

tristan3: [ontocomc.SubgroupIIb3, ontocomc.RootPositionChord] (1, ontocomc.e) (2, ontocomc.gh)

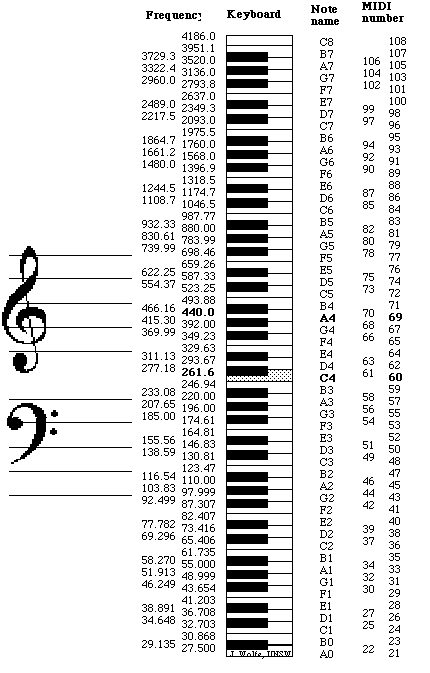
salient dissonance: ontocomc.tristan3\_23, [ontocomc.ChordInterval23, ontocomc.Tritone] >>> [ontocomc.ChordInterval23, ontocomc.Tritone]

tristan4: [ontocomc.SubgroupIIa] (1, ontocomc.e) (2, ontocomc.gh)

salient dissonance: ontocomc.tristan4\_23, [ontocomc.ChordInterval23, ontocomc.Tritone] >>> None

Several characteristics of Hindemith’s harmonic theory may be traced here: functionalities in relation to a tonality are irrelevant in this context, the focus being instead on the potential of the chords as such and on their dynamics within a sequence. All four chords belong to Group-B, indicating a lack of repose throughout. There is a clear inflexion at the third chord, as indicated by the resolution of the tritone of the previous chord to a minor 6th, while previous tritones remain unresolved to further tritones; this corresponds to the moment of landing on E in the bass. The harmonic fluctuation indicates a constant heightened tension, somewhat increasing in the two middle chords and relatively releasing on the last chord (in Subgroup IIa, corresponding to a tonal dominant 7th chord in root position).

# Appendix V: The piano range in MIDI pitch numbers



# Appendix VI: Herzliebster Jesu (phrase 1), chord analysis

*chords = []*

*for x in chorale\_by\_quaver[0][:24].index:*

*chord = hind.Chord(name='herzliebster'+'\_'+str(x),\*

*note4=chorale\_by\_quaver[0][:24].loc[(x,'pitchclass')],\*

*note3=chorale\_by\_quaver[1][:24].loc[(x,'pitchclass')],\*

*note2=chorale\_by\_quaver[2][:24].loc[(x,'pitchclass')],\*

*note1=chorale\_by\_quaver[3][:24].loc[(x,'pitchclass')])*

*chords.append(chord)*

*for i,chord in enumerate(chords[:-1]):*

*chord.nextChord = chords[i+1]*

*chords[-1].nextChord = hind.chordNil*

*hind.herzliebster = hind.ChordSequence(name='herzliebster',firstChord=chords[0])*

*hind.herzliebster.seq\_analysis()*

Hindemith's Craft of Musical Composition

Harmonic analysis of chord sequence 'herzliebster':

[ontocomc.eb, ontocomc.gb, ontocomc.a, ontocomc.b]

[ontocomc.eb, ontocomc.gb, ontocomc.a, ontocomc.b]

[ontocomc.e, ontocomc.e, ontocomc.g, ontocomc.b]

[ontocomc.e, ontocomc.e, ontocomc.g, ontocomc.b]

[ontocomc.f, ontocomc.d, ontocomc.ab, ontocomc.b]

[ontocomc.f, ontocomc.d, ontocomc.ab, ontocomc.b]

[ontocomc.gb, ontocomc.db, ontocomc.gb, ontocomc.bb]

[ontocomc.gb, ontocomc.db, ontocomc.gb, ontocomc.bb]

[ontocomc.gb, ontocomc.bb, ontocomc.db, ontocomc.gb]

[ontocomc.gb, ontocomc.bb, ontocomc.db, ontocomc.gb]

[ontocomc.ab, ontocomc.e, ontocomc.b, ontocomc.b]

[ontocomc.ab, ontocomc.e, ontocomc.b, ontocomc.b]

[ontocomc.bb, ontocomc.e, ontocomc.gb, ontocomc.db]

[ontocomc.bb, ontocomc.e, ontocomc.gb, ontocomc.db]

[ontocomc.b, ontocomc.d, ontocomc.gb, ontocomc.d]

[ontocomc.b, ontocomc.e, ontocomc.gb, ontocomc.d]

[ontocomc.b, ontocomc.gb, ontocomc.b, ontocomc.d]

[ontocomc.b, ontocomc.gb, ontocomc.b, ontocomc.d]

[ontocomc.bb, ontocomc.g, ontocomc.db, ontocomc.e]

[ontocomc.bb, ontocomc.g, ontocomc.db, ontocomc.e]

[ontocomc.b, ontocomc.gb, ontocomc.b, ontocomc.d]

[ontocomc.b, ontocomc.gb, ontocomc.b, ontocomc.d]

[ontocomc.gb, ontocomc.gb, ontocomc.bb, ontocomc.db]

[ontocomc.gb, ontocomc.gb, ontocomc.bb, ontocomc.db]

running the reasoner to classify intra-chord intervals for all chords...

running the reasoner to classify intervals between roving chord notes and the neighbor degree...

deroved chord 'ontocomc.herzliebster\_11', degree is 'ontocomc.b'

deroved chord 'ontocomc.herzliebster\_12', degree is 'ontocomc.b'

deroved chord 'ontocomc.herzliebster\_25', degree is 'ontocomc.e'

deroved chord 'ontocomc.herzliebster\_26', degree is 'ontocomc.e'

herzliebster\_7: [ontocomc.InversionChord, ontocomc.SubgroupIIb2] (4, ontocomc.b) (1, ontocomc.eb)

salient dissonance: ontocomc.herzliebster\_7\_13, [ontocomc.ChordInterval13, ontocomc.Tritone] >>> [ontocomc.ChordInterval13, ontocomc.Tritone]

herzliebster\_8: [ontocomc.InversionChord, ontocomc.SubgroupIIb2] (4, ontocomc.b) (1, ontocomc.eb)

salient dissonance: ontocomc.herzliebster\_8\_13, [ontocomc.ChordInterval13, ontocomc.Tritone] >>> [ontocomc.ChordInterval13, ontocomc.Min3rd]

herzliebster\_9: [ontocomc.SubgroupI1] (1, ontocomc.e) (1, ontocomc.e)

herzliebster\_10: [ontocomc.SubgroupI1] (1, ontocomc.e) (1, ontocomc.e)

herzliebster\_11: [ontocomc.SubgroupVI, ontocomc.InversionChord] (4, ontocomc.b) (0, ontocomc.pitchClassUndef)

salient dissonance: ontocomc.herzliebster\_11\_23, [ontocomc.Tritone, ontocomc.ChordInterval23] >>> [ontocomc.Tritone, ontocomc.ChordInterval23]

herzliebster\_12: [ontocomc.SubgroupVI, ontocomc.InversionChord] (4, ontocomc.b) (0, ontocomc.pitchClassUndef)

salient dissonance: ontocomc.herzliebster\_12\_23, [ontocomc.Tritone, ontocomc.ChordInterval23] >>> [ontocomc.Perf4th, ontocomc.ChordInterval23]

herzliebster\_13: [ontocomc.SubgroupI1] (1, ontocomc.gb) (1, ontocomc.gb)

herzliebster\_14: [ontocomc.SubgroupI1] (1, ontocomc.gb) (1, ontocomc.gb)

herzliebster\_15: [ontocomc.SubgroupI1] (1, ontocomc.gb) (1, ontocomc.gb)

herzliebster\_16: [ontocomc.SubgroupI1] (1, ontocomc.gb) (1, ontocomc.gb)

herzliebster\_17: [ontocomc.SubgroupI2] (2, ontocomc.e) (2, ontocomc.e)

herzliebster\_18: [ontocomc.SubgroupI2] (2, ontocomc.e) (2, ontocomc.e)

herzliebster\_19: [ontocomc.InversionChord, ontocomc.SubgroupIIb2] (3, ontocomc.gb) (1, ontocomc.bb)

salient dissonance: ontocomc.herzliebster\_19\_12, [ontocomc.ChordInterval12, ontocomc.Tritone] >>> [ontocomc.ChordInterval12, ontocomc.Tritone]

herzliebster\_20: [ontocomc.InversionChord, ontocomc.SubgroupIIb2] (3, ontocomc.gb) (1, ontocomc.bb)

salient dissonance: ontocomc.herzliebster\_20\_12, [ontocomc.ChordInterval12, ontocomc.Tritone] >>> [ontocomc.ChordInterval12, ontocomc.Min3rd]

herzliebster\_21: [ontocomc.SubgroupI1] (1, ontocomc.b) (1, ontocomc.b)

herzliebster\_22: [ontocomc.SubgroupIII1] (1, ontocomc.b) (1, ontocomc.b)

salient dissonance: ontocomc.herzliebster\_22\_24, [ontocomc.Min7th, ontocomc.ChordInterval24] >>> [ontocomc.ChordInterval24, ontocomc.Min6th]

herzliebster\_23: [ontocomc.SubgroupI1] (1, ontocomc.b) (1, ontocomc.b)

herzliebster\_24: [ontocomc.SubgroupI1] (1, ontocomc.b) (1, ontocomc.b)

herzliebster\_25: [ontocomc.SubgroupVI, ontocomc.InversionChord] (4, ontocomc.e) (0, ontocomc.pitchClassUndef)

salient dissonance: ontocomc.herzliebster\_25\_23, [ontocomc.Tritone, ontocomc.ChordInterval23] >>> [ontocomc.Tritone, ontocomc.ChordInterval23]

herzliebster\_26: [ontocomc.SubgroupVI, ontocomc.InversionChord] (4, ontocomc.e) (0, ontocomc.pitchClassUndef)

salient dissonance: ontocomc.herzliebster\_26\_23, [ontocomc.Tritone, ontocomc.ChordInterval23] >>> [ontocomc.Perf4th, ontocomc.ChordInterval23]

herzliebster\_27: [ontocomc.SubgroupI1] (1, ontocomc.b) (1, ontocomc.b)

herzliebster\_28: [ontocomc.SubgroupI1] (1, ontocomc.b) (1, ontocomc.b)

herzliebster\_29: [ontocomc.SubgroupI1] (1, ontocomc.gb) (1, ontocomc.gb)

herzliebster\_30: [ontocomc.SubgroupI1] (1, ontocomc.gb) (1, ontocomc.gb)

# References

1. **Hindemith, Paul.** *A Concentrated Course in Traditional Harmony.* s.l. : Schott, 1943/68.

2. —. *Craft of Musical Composition, Book I.* s.l. : Associated Music Publishers, Inc. , 1937.

3. **Halliday, John R.** *Paul Hindemith, the Theorist. PhD dissertation.* Rochester : Eastman School of Music, 1941.

4. **O' Connell, Kevin.** Hindemith's voices. *The Musical Times.* 2011.

5. **Hindemith, Paul.** *A Composer's World, horizons and limitations.* Cambridge : Harvard University Press, 1953.

6. **Drummond, Nick, et al.** *Putting OWL in Order: Patterns for Sequences in OWL.*

7. **Lamy, Jean-Baptiste.** *Ontologies with Python, Programming OWL 2.0 ontologies with Python and Owlready2.* s.l. : Apress, 2021.

8. **Kœchlin, Charles.** *Étude sur le Choral d’ École d’ après J. S. Bach.* Heugel. Paris : s.n., 1929.

9. **Bach, Johann Sebastian.** *371 vierstimmige Choräle.* [ed.] J. P. Kirnberger and C. P. E. Bach. s.l. : Breitkopf, 1787.

10. **Calomiris, Manolis.** *Harmony.* Athens : Gaitanos Music Publishing, 1933. Vol. I, in Greek.

11. **Damschroder, David.** *Thinking About Harmony.* s.l. : Cambridge University Press, 2008.

12. **Schoenberg, Arnold.** *Harmonielehre.* s.l. : Universal Edition, 1911/22.

13. **Riemann, Hugo.** *Harmony Simplified, the theory of the tonal functions of chords.* London : Augener, 1896.

14. **Dubois, Théodore.** *Traité d’ Harmonie Théorique et Pratique.* Paris : Heugel, 1921.

15. **Kœchlin, Charles.** *Étude sur les Notes de Passage.* Paris : Max Eschig, 1922.

16. **Searle, Humphrey.** *20th Century Counterpoint.* 2nd. London : Ernest Benn, 1955.

17. **Aldwell, Edward and Schachter, Carl.** *Harmony and Voice leading.* 3rd. s.l. : Schirmer, 2002.

18. **Allemang, Dean and Hendler, James.** *Semantic Web for the Working Ontologist.* [ed.] Morgan Kaufmann. 2η. 2011.

19. **Association, MIDI.** https://www.midi.org/specifications-old/item/the-midi-1-0-specification. [Online]

20. **Ebcioğlu, Kemal.** An expert system for harmonizing chorales in the style of J. S. Bach. *The Journal of Logic Programming.* 1990.

21. —. An expert system for harmonizing four-part chorales. [ed.] Stephan M. Schwanauer and David A. Levitt. *Machine Models of Music.* s.l. : MIT Press, 1993.

22. **Fang, Alexander, et al.** Bach or Mock?, A grading function for chorales in the style of J. S. Bach. *ICML.* 2020.

23. **Guarino, Nicola, Oberle, Daniel and Staab, Steffen.** What is an ontology? [ed.] Steffen Staab and Rudi Studer. *Handbook of Ontologies.* 2nd. s.l. : Springer, 2009.

24. **Hadjeres, Gaëtan, Pachet, François and Nielsen, Frank.** DeepBach, A steerable model for Bach chorales generation. *ISML.* 2017.

25. **Hertel, Alice, Broekstra, Jeen and Stuckenschmidt, Heiner.** RDF storage and retrieval systems. [ed.] Steffen Staab and Rudi Studer. *Handbook of Ontologies.* 2nd. s.l. : Springer, 2009.

26. **Hitzler, Pascal, Kroetzsch, Markus and Rudolph, Sebastian.** *Foundations of Semantic Web Technologies.* s.l. : CRC Press, 2010.

27. **Horridge, Matthew.** *A practical guide to building OWL ontologies using Protégé 4 and CO-ODE tools.* s.l. : University of Manchester, 2011.

28. **Kœchlin, Charles.** *Traité de l' Harmonie.* Paris : Max Eschig, 1928.

29. **Lamy, Jean-Baptiste.** *Owlready2 documentation, Release 0.23.* 2020.

30. **Rohrmeier, Martin.** Towards a generative syntax of tonal harmony. *Journal of Mathematics and Music.* 2011.

31. **Mickelsen, William C.** *Hugo Riemann's Theory of Harmony.* s.l. : University of Nebraska Press, 1977.

32. **Schoenberg, Arnold.** *Structural Functions of Harmony.* s.l. : Norton and Company, 1948/54.

33. **Kroetzsch, Markus.** *Description Logic Rules.* s.l. : AIFB, Karlsruher Institut fuer Technologie, 2010. D.Sc. Thesis.

34. **Lamy, Jean-Baptiste.** Owlready: ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. *Artificial Intelligence in Medicine.* 2017, 80.

35. **Maler, Wilhelm.** *Beitrag zur durmolltonalen Harmonielehre.* Athens : Nassos Ed., 1931/1983. in Greek.

36. **Stamou, George.** *Ontological Knowledge Representation and Reasoning.* s.l. : Kallipos, 2015. in Greek.

37. **Raimond, Yves, et al.** The Music Ontology. *Proceedings of the International Conference on Music Information Retrieval (ISMIR).* 2007.

38. **Musen, M. A.** The Protégé Project: A Look Back and a Look Forward. *Association of Computing Machinery Specific Interest Group in Artificial Intelligence.* 2015, Vol. 1(4).

39. **Pardo, Bryan and Birmingham, William P.** *The Chordal Analysis of Tonal Music.* s.l. : Unknown Publisher, 2001.

1. Composer, viola soloist and conductor, born in 1895 near Frankfurt, a leading figure in 20th century music. Extremely prolific, he composed operas, symphonic music, concertos, chamber music etc. Abandoning Nazi Germany in the 30s, he emigrated to USA and taught at Yale and elsewhere. He returned to Europe after the war, continued composing and resumed teaching in Zurich, where he lived until his death in 1963. Apart from *The* *Craft of Musical Composition*, he has notably written *Traditional Harmony* (in 2 volumes) and *Elementary Training for Musicians*. [↑](#footnote-ref-2)
2. Combination tones (or difference or Tartini tones) are barely perceptible bass frequencies when two tones sound together. A violinist may hear them as soft bass tones when he plays double stops perfectly in tune and organ manufacturers make use of them in small organs to produce a low tone with two smaller pipes in place of a large one. (3) [↑](#footnote-ref-3)
3. We are here introducing this term to denote the intervals formed between two tones belonging to the same chord. [↑](#footnote-ref-4)
4. Hindemith avoids using the term *inversion* altogether. In the following, we will be using both *inversion* and *root* *position* (whenever the root is in the bass), always bearing in mind that their meaning is here totally different from their tonal counterparts. [↑](#footnote-ref-5)
5. The latter chords correspond to our familiar diminished triads and diminished seventh chords, while the former correspond to augmented triads of tonal harmony. In CofMC, Subgroup V also includes chords consisting of two super-imposed perfect fourths, but this detail has been disregarded in this paper. [↑](#footnote-ref-6)
6. It is important to note here that the guide-tone is *not* a feature of some *key* or mode (like the leading note in tonal theory) but an inherent characteristic of a tritone *chord*. This is another original invention of Hindemith’s, without any ‘tonal’ counterpart. [↑](#footnote-ref-7)
7. In a chorale setting, this will typically be the highest voice (soprano). [↑](#footnote-ref-8)
8. Hindemith compares the usage of the chord degree to that of the logarithm in mathematics. [↑](#footnote-ref-9)
9. Sharps are usually indicated by using suffices -s and -ss, for # and ## respectively. Nevertheless, as suffix -s is traditionally used to denote *flat* tonalities in German notation (e.g. As-dur for A flat major), its use may be somewhat confusing. We have therefore provided an alternative spelling with suffix -h (vaguely alluding to #). Both spellings are valid and equivalent in all respects: C# may equally well be represented by *:cs* and/or *:ch*. [↑](#footnote-ref-10)
10. It is necessary to explicitly declare enharmonically equivalent tones as identical individuals in order to allow SWRL rules with atoms *sameAs()* and/or *differentFrom()* in their premises to produce conclusions as intended, irrespective of occasional pitch class spellings. [↑](#footnote-ref-11)
11. Admittedly, with a grain of salt: Hindemith concedes that chords with their tones extremely spaced out may not behave in exactly the same way as their dense copies, but, apart from such outlier cases, he holds fast to his invariance claim. [↑](#footnote-ref-12)
12. We must point out that the terms *root* *position* and *inversion*, though borrowed from tonal harmony, have a different meaning in this context. An inverted chord, in our case, is one with the root in some higher voice, no matter which other tone replaces the root in the bass. Although Hindemith does not sanction the use of these ‘tonal’ terms, he does acknowledge the distinction as a further “criterion for the appraisal of chords. All chords in which the bass tone and the root are not identical are subordinate to chords whose other characteristics (root and chord-group qualities) they share, but in which the root and bass tone do coincide” (2). [↑](#footnote-ref-13)
13. As explained, Hindemith defines *framework* as the sequence of intervals formed by the two most prominent voices in each chord, typically between the soprano and the bass in four-part writing. With some abuse of terminology, we have improvised the term ‘frame’ to denote the same interval for one chord in line. [↑](#footnote-ref-14)
14. The concept of *dissonance*, although ignored in CofMC, is here borrowed from tonal harmony, referring in our case to minor and major 2nds, minor and major 7ths and tritones; corresponding Series-2 values lie in [7,12]. Dissonance in tonal context would also encompass all augmented and diminished intervals, but those intervals are always enharmonically equivalent to simpler ones (e.g. the augmented 6th is equivalent to diminished 7th) and therefore are not treated separately in CofMC. [↑](#footnote-ref-15)
15. One may also cite cases expressible via some DL axiom where SWRL fails. [↑](#footnote-ref-16)
16. Lamy JB. Olwready, ontology-oriented programming in Python with automatic classification and high-level constructs for biomedical ontologies. Artificial Intelligence in Medicine 2017; 80:11-29 http://www.lesfleursdunormal.fr/\_downloads/article\_owlready\_aim\_2017.pdf [↑](#footnote-ref-17)
17. Note the filling out of blank voices in the chord with individual *:pitchClassUndef* [↑](#footnote-ref-18)
18. In this paper, such chords are called roving, in a quite loose sense following Schoenberg’s terminology in his *Harmonielehre*. [↑](#footnote-ref-19)
19. Bach has composed several more pieces based on chorales, such as chorale preludes, organ partitas, orchestral works etc. In all, Bach has used more than 400 chorales in his compositions. [↑](#footnote-ref-20)
20. A quaver or eighth note is a musical note (or pause) played for one eighth the duration of a whole note (semibreve). It is therefore a time unit relative to other rhythmic values in the piece. Its duration is half that of a crotchet (or quarter note) and twice that of a semiquaver (or sixteenth). Bach chorales are typically composed at the harmonic pace of quaver values or longer. The quaver pace is typically used for contrapuntal filling and voice leading, while the use of semiquavers is rather occasional and generally reserved to ornamentation. The choice of a semiquaver quantization does make sense and is often preferred, but our use of quaver quantization in this paper permits the ontology not to get overly populated and to remain reasonably uncluttered, missing only a small amount of information in the process. [↑](#footnote-ref-21)
21. The chords, ripped of their non-chord tones, have been entered manually in the ontology. Quavers with occurring non-chord tones have been padded by reproducing neighboring main chords. [↑](#footnote-ref-22)
22. Interestingly enough, the G# in the bass at quaver 48 does indicate a relative respite before a further rise in the next bar (obviously due to the suspended D in the tenor). [↑](#footnote-ref-23)
23. In the first case, as we have seen, we have a voice-leading stretching over a longer period; in the second case, we have an appoggiatura in the tenor resolving in the next crotchet over simultaneous quaver passing notes in the contralto. [↑](#footnote-ref-24)
24. There exist diminished and augmented intervals apart from 4ths and 5ths, for example the diminished third A – C flat. These intervals are *enharmonically* equivalent to some ‘simpler’ interval (A – B, in our case) and normally arise as part of altered chords in tonal harmony. As we shall see, Hindemith does not consider such intervals as independent entities, therefore they will not concern us any further in this paper. [↑](#footnote-ref-25)
25. A chord sequence is typically viewed as a series of vertical simultaneities intended for a mixed chorus in 4 parts (sopranos, contraltos, tenors and basses, from the top voice downwards). [↑](#footnote-ref-26)
26. Considering modulations in a musical piece not as mere arbitrary key changes but as a fine interplay between tonal regions leads to a more integrated view of tonality; Schoenberg pushes this idea to the limit with his principle of monotonality: “…every digression from the tonic is considered to be still within the tonality, whether directly or indirectly, closely or remotely related. In other words, there is only one tonality in a piece, and every segment formerly considered as another tonality is only a region, a harmonic contrast within that tonality”. (32) [↑](#footnote-ref-27)
27. Certain such formations are routinely used in idioms beyond classical music and have acquired an independent harmonic status, e.g. suspended chords in jazz and rock music. [↑](#footnote-ref-28)
28. The adjective *notorious* would be equally well justified: to most listeners in the late 19th and well into the 20th century, this sequence sounded indeed like the death warrant of tonality. It has been extensively analyzed by dozens of experts over the years in several different and often contradictory ways, in a repeated attempt to deal with its extreme tonal vagueness. The first chord, its ambiguous sound banishing almost all tonal functionality, has in fact been dubbed as the ‘Tristan chord’. [↑](#footnote-ref-29)
29. This choice is further open to different annotations of the first chord, by enharmonically reinterpreting D# and G#. [↑](#footnote-ref-30)