City Research Online

City, University of London Institutional Repository

Citation: Wissmann, Jens, Weyde, T. & Conklin, D. (2010). Representing chord sequences in OWL. Paper presented at the Sound and Music Computing Conference 2010, 21 - 24 June 2010, Barcelona, Spain.

This is the unspecified version of the paper.

This version of the publication may differ from the final published version.

Permanent repository link: https://openaccess.city.ac.uk/id/eprint/2754/

Link to published version:

Copyright: City Research Online aims to make research outputs of City, University of London available to a wider audience. Copyright and Moral Rights remain with the author(s) and/or copyright holders. URLs from City Research Online may be freely distributed and linked to.

Reuse: Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

City Research Online: http://openaccess.city.ac.uk/ publications@city.ac.uk

CHORD SEQUENCE PATTERNS IN OWL

Jens Wissmann
FZI Research Center for Information Technologies,
Karlsruhe, Germany
jens.wissmann@fzi.de

Tillman Weyde
City University London,
London, United Kingdom
t.e.weyde@city.ac.uk

Darrell Conklin
Department of Computer Science and AI
Universidad del País Vasco,
San Sebastián, Spain
IKERBASQUE, Basque Foundation for Science
darrell_conklin@ehu.es

ABSTRACT

Chord symbols and progressions are a common way to describe musical harmony. In this paper we present $\mathcal{SEQ}$, a pattern representation using the Web Ontology Language OWL DL and its application to modelling chord sequences. $\mathcal{SEQ}$ provides a logical representation of order information, which is not available directly in OWL DL, together with an intuitive notation. It therefore allows the use of OWL reasoners for tasks such as classification of sequences by patterns and determining subsumption relationships between the patterns. The $\mathcal{SEQ}$ representation is used to express distinctive pattern obtained using data mining of multiple viewpoints of chord sequences.

1. INTRODUCTION

The Semantic Web is an effort to augment the conventional Web with explicit machine-processable semantic metadata to serve as a backbone for a variety of automated content processing and retrieval task [1, 2]. In this context, several techniques for the logical description and querying of web data have been developed. Particularly, modelling of knowledge in web ontologies using the Description Logic OWL DL [3] enables automatic reasoning. However, these techniques have been developed with the focus on terminological metadata and the use of these techniques to reason on structured objects such as found in music representation is still in its beginnings.

For our approach, we chose chord sequences as a starting point as these are a popular representation and have increasingly gained research interest [4, 5]. They are also at a convenient and powerful level of musical abstraction. For example, within the “Music Ontology" effort patterns have been learned from chord sequences available in the Semantic Web data format RDF [6, 7]. The patterns themselves however have not been expressed with Semantic Web techniques. Indeed, neither RDF nor OWL offer ad hoc support for representing sequential structures.

We have developed a generic representation for sequential patterns in OWL DL that we call $\mathcal{SEQ}$, extending the work of [8], and applied it to chord sequence representation. Notation and expressivity are similar to regular expressions, and allow the expression of different levels of abstraction. Several reasoning tasks on such a representation can be solved using readily available OWL reasoners. In a web retrieval scenario, for example, instance checking can be used to find chord sequences that match or contain a search pattern. More interestingly, subsumption checking analyses pattern inclusion.

To demonstrate how the $\mathcal{SEQ}$ representation can be used to enrich the results of pattern discovery, we translated distinctive chord patterns, which were learned from a corpus using a statistical learning approach in [9], into $\mathcal{SEQ}$ and used an OWL reasoner for the calculation of subsumption relations and instance retrieval.

Copyright: ©2010 Wissmann et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. MODELLING KNOWLEDGE IN OWL DL

OWL DL belongs to the Description Logic (DL) family of knowledge representation languages [10]. DLs are popular for describing the knowledge of a domain of interest by formalising its terminology using

- *instances* $i, j, \ldots$,
- *concepts* $C, D, \ldots$ and
- *properties* $R, S, \ldots$

Most DLs correspond to fragments of first order logic such that *instances*, *concepts* and *properties* correspond to *constants*, *unary predicates* and *binary predicates*.

An *ontology* is a set of axioms that define relationships between these terms. The part of the ontology that asserts facts about instances is called the ABox, while the part that defines the terminology is called TBox. From a first order logic perspective, ABox axioms assert predicates on constants while TBox axioms describe predicate structures on variables. Basic forms of terminological axioms are

- concept subsumption ($C \sqsubseteq D$) and
- equivalence ($C \equiv D$).

Basic forms of assertional axioms are

- type assertions ($i \in C$) and
- property assertions ($R(i, j)$).

Here C and D can stand for *atomic* concepts but can also be composite *expressions* as we will further illustrate.

In this paper we mainly focus on modelling structural aspects of chord sequences, but will consider some example concept expressions from the domain of music metadata as DL syntax was originally introduced for describing terminologies and it is therefore most intuitive to describe

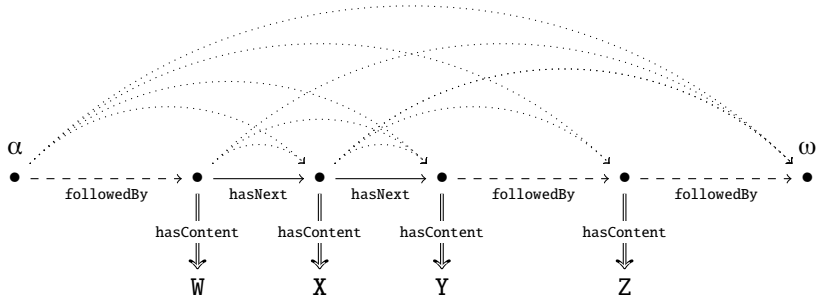

Figure 1: Structure of an example sequential pattern

the relationships between words. A motivation for this is also to highlight the possibilities of DLs for reasoning on musical structures and musical metadata within one single logical framework. For example, consider the TBox

$$\texttt{Musician} \equiv \exists \texttt{performed}.\texttt{Music} \sqcup \exists \texttt{wrote}.\texttt{Music}$$
$$\texttt{Composer} \equiv \exists \texttt{wrote}.\texttt{Music}$$

These axioms define a musician as somebody who performed or wrote music, and a composer to be someone who wrote music. Here boolean constructs and property restrictions are used to form expressions. DLs provide boolean constructors $\neg C$, $C \sqcap D$, $C \sqcup D$. As DLs have first order logic semantics we can think of these as complement, intersection and union of sets (of instances). Further, DLs allow to quantify over properties ($\exists R.C$, $\forall R.C$, $\exists^{=n} R.C$, $\exists^{\leq n} R.C$, $\exists^{\geq n} R.C$), e.g. stating that for an instance that is a `Composer` there exists a property `wrote` with the range `Music`. OWL Reasoners provide certain standard reasoning services. For example, by *subsumption reasoning* on the TBox a reasoner can be infer that all composers are necessarily musicians (`Composer` $\sqsubseteq$ `Musician`). In fact all subsumption problems in DLs are decidable, i.e. we can do this for any two concept descriptions. So the main challenge is to capture the interesting aspects of a terminology as DL axioms, whereas the reasoning is done automatically.

A further reasoner task is classification of an ABox with respect to TBox concepts. Consider the facts

$$\texttt{wrote}(\texttt{mozart}, \texttt{magic\_flute}),$$
$$\texttt{wrote}(\texttt{shakespeare}, \texttt{hamlet}),$$
$$\texttt{magic\_flute} \in \texttt{Music},$$
$$\texttt{shakespeare} \in \forall \texttt{wrote}.\texttt{Literature}$$

Here, for example, Mozart will be classified as composer and musician. Shakespeare will not be classified as musician as he just wrote literature.

Additional DL constructs exist that allow to assert subproperty relationship, inverse property relationship and characteristics of properties such as being functional, transitive, reflexive, irreflexive, symmetric or asymmetric. We refer the reader to [10] and [11] for a more detailed discussion of DLs.

3. MODELLING SEQUENCES IN OWL

The wish to model sequences arises naturally in the music domain, given its temporal nature. Unfortunately, there are no native constructs within OWL DL to express sequence patterns. Drummond et al. [8] proposed to use a *linked list* approach. We extended this approach and developed $\mathcal{SEQ}$, an ontological representation of sequence patterns.

In the following we describe the axiomatisation of basic $\mathcal{SEQ}$ patterns and give examples. The axiomatization of the linked list structure follows the ideas of Drummond et al. [8]. One difference is that we introduce an initial component because this is crucial for the behaviour of pattern subsumption and for the creation of more complex pattern constructs. Further we introduce a notation to express sequences in a more intuitive (yet formal) way.

The core structure of a $\mathcal{SEQ}$ pattern is similar to a linked list. Figure 1 shows an example. Components of patterns are linked by solid dots. Each component can be associated with linking and content properties: Linking is expressed by using the *functional* property `hasNext` (solid arrow) that connects a component to its immediate successor or by using the *transitive* property `followedBy` (dashed arrow) that connects a component to all following components. A *pattern* is characterised by restricting these properties. As the subproperty relationship `hasNext` $\sqsubseteq$ `followedBy` is asserted for $\mathcal{SEQ}$ patterns, `followedBy` relationships are are implicitly defined between all connected components (dotted arrows).

The property `hasContent` can be used to describe the content of a pattern component. Finally, we introduce an intital component (α) with no precursor and no content and a final component (ω) with no successor and no content (see table 1 for definitions) . A sequence pattern $\texttt{SP}_1$ that describes sequences that consist of "some instances of `W`, then `X`, then `Y`, then followed by `Z`" (as shown in fig. 1) can be described by the DL concept

$$\begin{aligned}\texttt{SP}_1 \equiv \alpha \sqcap \exists \texttt{followedBy}.(&\exists \texttt{hasContent}.\texttt{W} \\ \sqcap \quad &\exists \texttt{hasNext}.(\exists \texttt{hasContent}.\texttt{X} \\ \sqcap \quad &\exists \texttt{hasNext}.(\exists \texttt{hasContent}.\texttt{Y} \\ \sqcap &\exists \texttt{followedBy}.(\exists \texttt{hasContent}.\texttt{Z} \\ \sqcap &\exists \texttt{followedBy}.\omega))))\end{aligned}$$

For simplification, we can state this expression equivalently

	Syntax	Semantics	
succeeds	$C \triangleright D$	$C \sqcap \exists \texttt{hasNext}.D$	TBox
follows	$C \cdots D$	$C \sqcap \exists \texttt{isFollowedBy}.D$	
has content	$[C]$	$\exists \texttt{hasContent}.C$	
initial	α	$\neg\exists \texttt{followedBy}^{-}.\top \sqcap \exists^{\leq 0}\texttt{hasContent}.\top$	
terminal	ω	$\neg\exists \texttt{followedBy}.\top \sqcap \exists^{\leq 0}\texttt{hasContent}.\top$	

Table 1: A selection of $\mathcal{SEQ}$ constructs and their definition. C and D denote arbitrary DL concepts

in $\mathcal{SEQ}$ notation as

$$\texttt{SP}_1 \equiv [\texttt{W}] \triangleright [\texttt{X}] \triangleright [\texttt{Y}] \cdots [\texttt{Z}]$$

with α and ω are not explicitly stated and arrows, dots and square brackets capturing the details of the succeeds, follows and content restrictions. Consider, the pattern

$$\texttt{SP}_2 \equiv [\texttt{W}] \triangleright [\texttt{X}] \triangleright [\texttt{Y}] \triangleright [\texttt{Z}]$$

The difference with $\texttt{SP}_1$ here is that Y has to be directly followed by Z. Intuitively we expect that $\texttt{SP}_2$ is more specialized than $\texttt{SP}_1$ and all instances of $\texttt{SP}_2$ will also be instances of $\texttt{SP}_1$. As we have formalized $\mathcal{SEQ}$ patterns as DL concepts, we can directly use the machinery for computing DL concept subsumption to automatically compute pattern subsumption. In this case a standard DL reasoner will infer the subsumption relationship $\texttt{SP}_2 \sqsubseteq \texttt{SP}_1$ (taking into consideration that hasNext is a subproperty of followedBy).

The possibilities of subsumption reasoning get more interesting when we use concept *expressions* (such as we have done in the Musician example) rather than simple concept names. For example we could define a chord by its properties, e.g.

$$\begin{bmatrix} \exists \ \ \texttt{root} \ . \ \texttt{C} \\ \sqcap \exists \ \ \texttt{triad} \ . \ \texttt{Maj} \\ \sqcap \exists \ \texttt{seventh} \ . \ \texttt{b7} \end{bmatrix}$$

where the pattern characterises a chord by the properties root, triad and seventh. Given another more general pattern that for example only restricts root and triad a reasoner could infer a subsumption relationship such as

$$\begin{bmatrix} \exists \ \ \texttt{root} \ . \ \texttt{C} \\ \sqcap \exists \ \ \texttt{triad} \ . \ \texttt{Maj} \\ \sqcap \exists \ \texttt{seventh} \ . \ \texttt{b7} \end{bmatrix} \sqsubseteq \begin{bmatrix} \exists \ \ \texttt{root} \ . \ \texttt{C} \\ \sqcap \exists \ \texttt{triad} \ . \ \texttt{Maj} \end{bmatrix}$$

In the work described in the following section we restrict ourselves to patterns that describe their content as a conjunction of features (functional properties) as we can discover patterns of this form automatically using the pattern discovery method by [9]. Note, that in principle it is also possible to make use of further DL operators when defining patterns. For example, the pattern

$$\begin{bmatrix} \exists \ \ \texttt{root} \ . \ \neg(\texttt{F} \sqcup \texttt{G}) \\ \sqcap \exists \ \texttt{triad} \ . \ \texttt{Maj} \end{bmatrix}$$

matches major chords that have a root other then F or G, and given our previous example pattern would give rise to the subsumption relationship:

$$\begin{bmatrix} \exists \ \ \texttt{root} \ . \ \texttt{C} \\ \sqcap \exists \ \ \texttt{triad} \ . \ \texttt{Maj} \\ \sqcap \exists \ \texttt{seventh} \ . \ \texttt{b7} \end{bmatrix} \sqsubseteq \begin{bmatrix} \exists \ \ \texttt{root} \ . \ \neg(\texttt{F} \sqcup \texttt{G}) \\ \sqcap \exists \ \texttt{triad} \ . \ \texttt{Maj} \end{bmatrix}$$

Naturally the question arises how such patterns can be created in practise. As manual modelling is often costly and time consuming, it is interesting to investigate methods for automatic pattern creation. In the following section we will outline the relationship of the $\mathcal{SEQ}$ formalism to the established viewpoint approach to automatic pattern discovery.

4. SUBSUMPTION STRUCTURE OF DISTINCTIVE CHORD PATTERNS

Though $\mathcal{SEQ}$ patterns can be specified in a top-down manner by a knowledge engineer, it is interesting to learn them from a corpus of music. This approach leads to the question which patterns are most relevant and interesting, which is a typical question from the field of data mining. Depending on the application, there are different relevant properties. For the classification of music, which is very useful in a Semantic Web scenario, we are interested in distinctive patterns that help differentiate one class from another, and general patterns that apply to many relevant data sets in a class. Conklin [9] has applied this approach to chord sequences and found a number of relevant patterns that we further analysed using $\mathcal{SEQ}$.

4.1 Representation of Feature Set Patterns

Pattern discovery using *multiple viewpoints* is a machine learning approach for discovering patterns in sequential musical data. It has mainly been used for discovery of patterns in melodies, but recently also for learning patterns in chord progressions [9]. Input and patterns are represented using a *feature set* representation [12].

For a sequence of musical events (e.g. chords), viewpoints are computed. A viewpoint τ is a function from events to values in a specific range set. A feature is defined as $\tau : v$ where τ is a feature name and v a feature value. A *feature set* then is a *conjunction* of features

$$\{\tau_1 : v_1, \ldots, \tau_n : v_n\}$$

and a *pattern* is a sequence

$$f_1, \ldots, f_m$$

where each f_i is a feature set.

	events:	Im7	IVm7	Im7	Vbm7b5	IV7	IIIsm7b5
features	degree	I	IV	I	Vb	IV	IIIs
	basedegree	I	IV	I	V	IV	III
	kp	I	II/IV	I	V/VII	II/IV	III
	triad	Min	Min	Min	Dim	Maj	Dim
	rootmvt	⊥	4n	5n	5b	7s	7n

Table 2: Example decomposition of chord-events into feature sets for viewpoint learning

Table 2 shows an example of how a chord progression is represented as a sequence of feature sets. The viewpoints `degree`, `triad` and `basedegree` directly relate to the chord symbol. Relationships between events are modelled as features that belong to a single event and have to be read as referring back to the previous event. The feature `rootmvt : 4n` for example expresses that the current root event is a fourth about the previous event. In the case of the first event features of this kind take the value ⊥ as there is no previous event they could refer to. We use further viewpoints in later examples such as `meeus` that indicates harmonic function (tonic (`T`), dominant (`D`) or subdominant (`S`)) as described by [13], `kp` that indicates chord degree classes as described by [14] and `ratio(dur)` that indicates the relative duration of an event.

4.2 Translation of Feature Set Patterns to $\mathcal{SEQ}$

Feature set patterns can be translated into $\mathcal{SEQ}$ using a translation function T that is defined as follows. Each feature $\tau : v$ can be translated into a DL property restriction

$$T(\tau : v) = \exists \tau.v$$

where every viewpoint τ corresponds to a *functional* property τ and the value v is the filler that the property is restricted to. A feature set is described by a DL concept intersection

$$T(\{\tau_1 : v_1, \ldots, \tau_n : v_n\}) = \exists \tau_1.v_1 \sqcap \ldots \sqcap \exists \tau_n.v_n$$

A feature set pattern $f_1 \ldots f_m$ can then be expressed using `hasNext` relationships as

$$T(f_1, \ldots, f_m) = [T(f_1)] \triangleright \ldots \triangleright [T(f_m)]$$

In the following we will show examples of genre-specific chord sequence patterns that have been learned from chord sequences tagged with the genres *jazz*, *classic* and *pop*.

4.3 Maximally General Distinctive Chord Patterns

A *maximally general distinctive pattern* (MGDP) is a pattern that is distinctive above a threshold and not subsumed by any other distinctive pattern. They are least likely to overfit the corpus and hence most likely to be useful for classification. To measure distinctiveness the likelihood ratio of a pattern P is employed. This is defined in [9, 15] as

$$\Delta(P) \stackrel{def}{=} \frac{p(P|\oplus)}{p(P|\ominus)} = \frac{c^{\oplus}(P) \times n^{\ominus}}{c^{\ominus}(P) \times n^{\oplus}}$$

where $p(P|\oplus)$ is the probability of the pattern P in the corpus, $p(P|\ominus)$ is the probability of the pattern P in the anticorpus (consisting of pieces of different classes), $c^{\oplus}(P)$ and $c^{\ominus}(P)$ are the count of the pattern in the corpus and the anticorpus respectively, and $n^{\oplus}$ and $n^{\ominus}$ are the size of the corpus and anticorpus respectively.

Figure 2 (top) illustrates three MGDPs chosen from a much larger set of highly distinctive patterns that were discovered in a corpus of 856 chord sequences, divided into genres jazz (338), classical (235), and popular (283) [16]. The interest $\Delta(P)$ of the pattern is indicated: for example, the first pattern is overrepresented by a factor of 12.45. The numbers in brackets indicate that the length of the pattern is 2 and it occurs in 65 jazz sequences but only 8 sequences in the anticorpus (classical and popular sequences). The pattern indicates a minor triad on degree `III`, followed by any triad on degree `III` (due to the fact that the `meeus` property indicates the `T` (tonic) chord transformation). Note that despite this high level of abstraction in this pattern it remains highly distinctive in this corpus for the jazz genre.

In the middle of Figure 2, instances of each of these patterns are represented as fully saturated feature set sequences.

4.4 Subsumption Structure

To compute the subsumption structure of the learned viewpoint patterns we translated viewpoint patterns into $\mathcal{SEQ}$ concepts and used a DL reasoner to infer their subsumption relationships.

The bottom part of Figure 2 illustrates a small fragment of a subsumption hierarchy of viewpoint patterns, created from a larger set of pattern that are *maximally general* and *distinctive* (MGDP). The subsumption relationships were computed by the $\mathcal{SEQ}$-translation of the MGDPs. To compute the subsumption relationships we translated the patterns into $\mathcal{SEQ}$ and then use the OWL reasoner Pellet [1] to classify. This figure has restricted the representation to five MGDP that appear on the righthand side of the hierarchy.

Some internal concepts have been constructed in $\mathcal{SEQ}$ and it can be seen how these capture commonalities between the MGDPs thereby providing richer structure to a flat MGDP set. At the left hand side of the figure are "primitive" features contained in single component patterns. Substantial structure can be seen. For example, `triad : Min` can be seen to occur in four MGDPs and in addition in one internal $\mathcal{SEQ}$ pattern.

[1] http://clarkparsia.com/pellet/

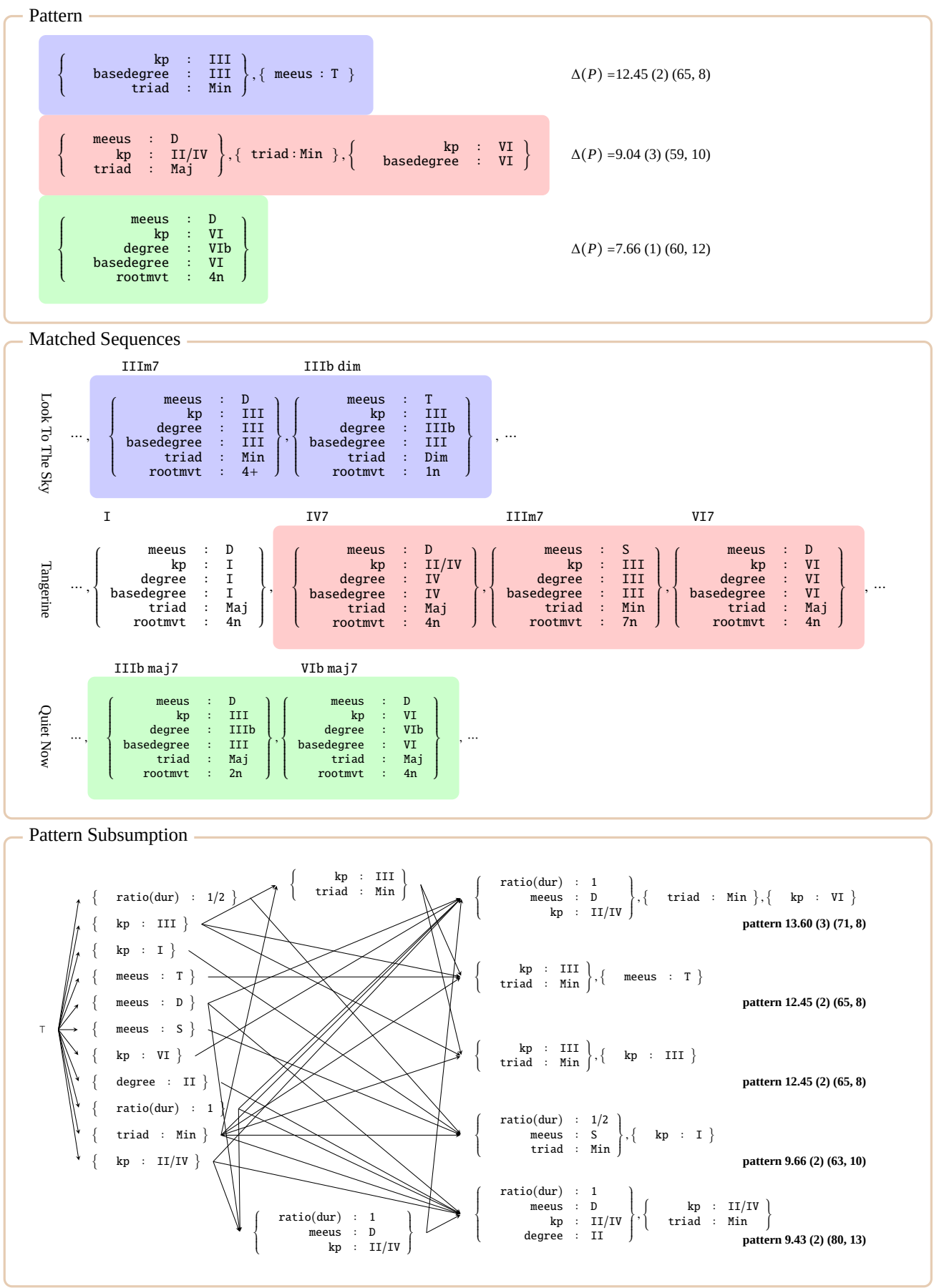

Figure 2: Example of learned MGDP-patterns (top), matching sequences (middle) and pattern subsumption (bottom).

5. CONCLUSIONS

We introduced the $\mathcal{SEQ}$ language and showed how it expresses sequential patterns and discussed some aspects of syntax and the DL semantics of $\mathcal{SEQ}$. We demonstrated the usage of $\mathcal{SEQ}$ to represent and analyse chord patterns that were discovered from a corpus using viewpoint learning. A DL reasoner can then use such patterns to classify instance data. Further, the patterns can be classified automatically in terms of their subsumption relationships as illustrated for distinctive patterns from [9].

Several possibilities for future research arise. Reasoning on metadata descriptions (as in our introductory example) and structural descriptions within the same reasoning formalism might offer interesting new application possibilities for musicology and music information retrieval. Further, the machine-learned descriptions could be complemented with relationships between basic musical entities such as notes, scales and chord as found in the harmony literature.

6. REFERENCES

[1] T. Berners-Lee, *Weaving the Web : the past, present and future of the World Wide Web by its inventor*. London: Orion Business, 1999.

[2] T. Berners-Lee, J. Hendler, and O. Lassila, "The SemanticWeb," *Scientific American*, vol. 284, pp. 34–43, May 2001.

[3] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph, eds., *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-primer/`.

[4] C. Harte, M. B. Sandler, S. A. Abdallah, and E. Gómez, "Symbolic representation of musical chords: A proposed syntax for text annotations," in *ISMIR*, pp. 66–71, 2005.

[5] A. Sheh and D. P. W. Ellis, "Chord segmentation and recognition using em-trained hidden markov models.," in *ISMIR*, 2003.

[6] A. Anglade and S. Dixon, "Characterisation of harmony with inductive logic programming," in *Proc. of the Ninth International Conference on Music Information Retrieval (ISMIR)*, (Philadelphia, USA), pp. 63–68, Sep 2008.

[7] M. Mauch, S. Dixon, C. Harte, M. Casey, and B. Fields, "Discovering chord idioms through Beatles and Real Book songs," in *Proceedings of ISMIR 2007 Vienna, Austria*, pp. 255–258, 2007.

[8] N. Drummond, A. Rector, R. Stevens, G. Moulton, M. Horridge, H. H. Wang, and J. Seidenberg, "Putting OWL in Order: Patterns for Sequences in OWL," in *2nd OWL Experiences and Directions Workshop, Athens, GA*, 2006.

[9] D. Conklin, "Discovery of distinctive patterns in music," To appear in *Intelligent Data Analysis*, vol. 14, no. 5, 2010.

[10] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, eds., *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[11] F. Baader, I. Horrocks, and U. Sattler, "Description Logics," in *Handbook of Knowledge Representation* (F. van Harmelen, V. Lifschitz, and B. Porter, eds.), Elsevier, 2007.

[12] D. Conklin and M. Bergeron, "Feature set patterns in music," *Computer Music Journal*, vol. 32, no. 1, pp. 60–70, 2008.

[13] N. Meeus, "Toward a post-Schoenbergian grammar of tonal and pre-tonal harmonic progressions," *Music Theory Online*, vol. 6, January 2000.

[14] S. Kostka and D. Payne, *Tonal Harmony*. McGraw-Hill, 2003.

[15] D. Conklin, "Distinctive Patterns in the First Movement of Brahms's String Quartet in C Minor," To appear in *Journal of Mathematics and Music*, vol. 4, no. 2, 2010.

[16] C. Pérez-Sancho, D. Rizo, and J.-M. Iñesta, "Genre classification using chords and stochastic language models," *Connection Science*, vol. 20, no. 2&3, pp. 145–159, 2009.