AN INITIAL COMPUTATIONAL MODEL FOR MUSICAL SCHEMATA THEORY

Andreas Katsiavalos[1] **Tom Collins**[2,3] **Bret Battey**[1]
[1] Music, Technology and Innovation Research Centre, De Montfort University, UK
[2] Music, Science and Technology Research Cluster, Department of Music, University of York, UK
[3] Music Artificial Intelligence Algorithms, Inc., Davis, CA, USA
andreas.katsiavalos@gmail.com, tomthecollins@gmail.com

ABSTRACT

Musical schemata theory entails the classification of subphrase-length progressions in melodic, harmonic and metric feature-sets as named entities (e.g., 'Romanesca', 'Meyer', 'Cadence', etc.), where a musical schema is characterized by factors such as music content and form, position and tonal function within phrase structure, and interrelation with other schemata. To examine and automate the task of musical schemata classification, we developed a novel musical schemata classifier. First, we tested methods for exact and approximate matching of user-defined schemata prototypes, to establish the notions of identity and similarity between composite music patterns. Next, we examined methods for schemata prototype extraction from collections of same-labelled annotated examples, performing training and testing sessions similar to supervised learning approaches. The performance of the above tasks was verified using the same annotated dataset of 40 keyboard sonata excerpts from pre-Classical and Classical periods. Our evaluation of the classifier sheds light on: (a) ability to parse and interpret music information, (b) similarity methods for composite music patterns, (c) categorization methods for polyphonic music.

1. INTRODUCTION

Schemata have been characterised by psychologists as 'the building blocks of cognition' [17], enabling an understanding of the world in packages of knowledge. Similarly, musical schemata can be thought of as 'minimal meaningful' entities, enabling coherent interpretations of Classical phrases. Musical schemata theory is studied and developed by musicologists as a means of classifying short passages in musical works, mainly from the Classical period [2, 10].

Aiming to model musical schemata theory, we consider the development of computational systems that create and update definitions for prototypes of schemata categories from annotated music examples. This work builds on a relatively small amount of existing research in computational modeling of musical schemata theory [8, 9, 18].

The novel system will enable the study of higher-level operations and reasoning in content-based music information retrieval than has been possible to date, and facilitate further research with machine learning approaches in music pattern extraction [15, 19, 20].

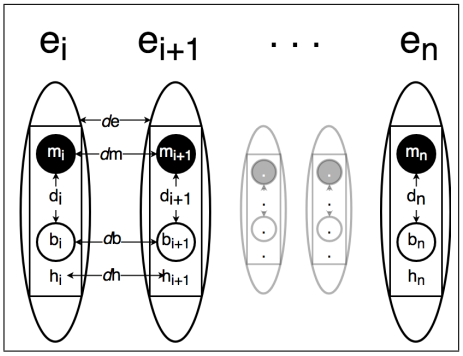

Figure 1. Musical schemata are considered as sequences of schema-events.

2. TASK DESCRIPTION AND BACKGROUND

2.1 Musical schemata in Classical keyboard works

A musical schema is defined as a stereotypical progression of *schema-event* elements (Figure 1): a feature-set consisting of notes from two melodic movements (melody and bass) and harmonic and metric information [7, 10]. A schema is characterized by its content, that is, the number and type of its constituent schema-events, but also as part of phrases and even greater morphological entities such as paragraphs, periods, etc., as well as its position, tonal function, and any interrelations with other schemata.

An interesting aspect of the theory is the notion of a prototype for a set of similar progressions. The extraction of schemata prototypes is a learning process and musical schemata theory is an example-based approach, meaning that knowledge about a schema prototype is obtained and updated from examples and not by rules.

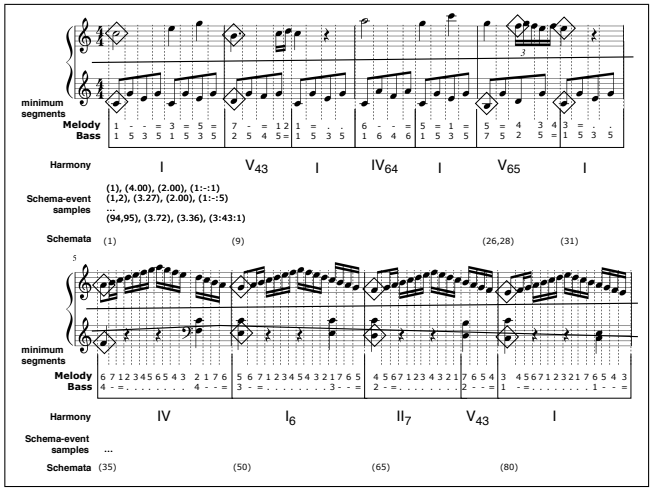

Figure 2. Annotated example with melodic movement separation (horizontal lines), harmonic regions, and schemata notes (diamonds). The first line of the score has a 'Meyer' instance and the second, a 'Prinner'. Excerpt from Wolfgang Amadeus Mozart, Piano Sonata no.16 in C major K545 1st movement Allegro mm.1-8.

2.2 Modeling musical schemata theory

We implemented example-based machine learning with the development of a musical schemata classifier. The classifier works in three steps: making observations of the music data, comparing these observations to stored schema prototypes, and identifying schema-class similarity.

2.2.1 Observations

To classify music patterns such as musical schemata, the input needs to be processed so that only relevant, 'schematic' material is considered. As described previously, musical schemata are viewed as progressions of schema-events, and for that reason we consider two types of 'observations': a) schema-states, and, b) schema-state combinations/progressions. We will refer to these elements as 'low-' and 'high-observations' respectively. The schema-states are schema-voice and schema-event samples, a kind of low-level form of music understanding for small feature-sets. Combining these schema-state elements, we can create schemata instances/samples that are comparable to musical schemata prototypes and thus, perform similarity and classification tasks.

Creating the schema-state observations is a preprocessing step, independent from the tasks of recognition and learning, but the observation process can utilize information from stored prototypes to select only known (stored) schema-states and, thus, reduce the number of schema-event state observations.

2.2.2 Similarity

A fundamental task of the classifier is to perform comparisons of same-class information (e.g. states or schemata) from different sources (e.g. observations, prototypes). The matching of observations from a source score and a stored schema prototype is the recognition process. To evaluate our musical schemata classifier, these recognition results are compared/validated against schemata annotations, a 'ground-truth', giving measures of recall and precision.

To perform comparisons, the idea is to define a similarity metric for multi-feature elements and sequences, such as schema-states and complete schemata, and utilize it to identify and categorize them according to their content.

2.2.3 Class similarity

In addition to recognition and evaluation similarity, class similarity is another type of similarity whereby common relations amongst schema states and schemata observations of the same schema type are identified, hence, describing the properties of a schema family type. In this study, a schema class is defined by a set of examples, an example-base of high-level observations, and a class (similarity) function that validates all its existing examples, acting as the identity validator for all of them.

An interesting aspect of our classification method for schemata is that, contrary to matching methods that create a search-space for specific targets, our approach can go beyond merely finding matches to existing schemata prototypes to automatically identifying potential schemata.

3. METHODOLOGY

3.1 Method overview

The musical schemata classifier was developed by application to incrementally more complex scenarios:

- Identification (labelled I hereafter). Initially, the classifier performed musical schemata identification with pattern matching techniques, identifying the exact positioning of user-defined schema-family prototypes in complete music parts;
- Recognition (labelled R and F). Next, we examined the task of multiple and approximate recognition of user-defined schema-family prototypes, to study similarity-approximation methods for both single-family prototypes (R) and collections of multiple variants for a single schema class (F);
- Learning (labelled L). Finally, we tested the schemata classifier on learning of schemata prototypes from schema-annotated music examples.

The annotations were of a dataset of 40 keyboard sonata parts (10 from Haydn, 10 from Mozart, and 20 from Beethoven), containing annotations for 3 schemata types ('Meyer', 'Prinner', and, 'Cadence'). In total the number of unique annotations are: 17 generic and 23 'Meyer' variations, 22 generic and 8 'Prinner' variations, and, 40 generic and 15 'Cadence' variations.

3.2 Model overview

The details of the model's implementation are beyond the scope of this paper and therefore the following provides

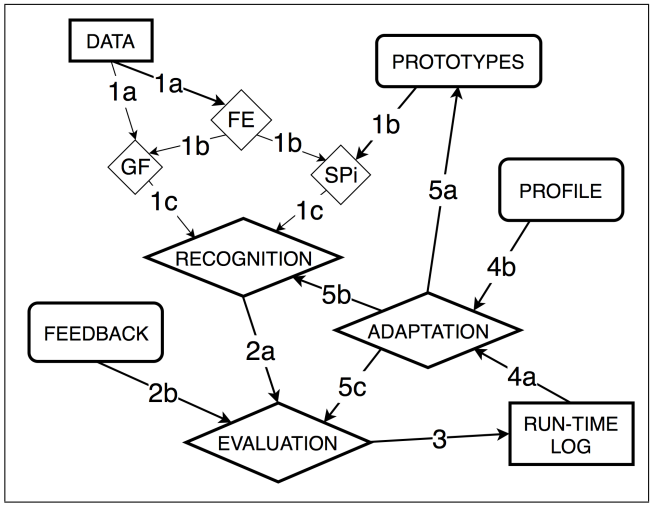

Figure 3. Top view of the schemata classifier model.

only a high-level overview of the approach. [1] The classifier runs in sessions where information from up to five sources is processed and combined (please see Figure 3). The *music-data* source (Figure 3, DATA) supplies complete sonata parts in MusicXML format sequentially, and each input file follows the same information processing path, where the input file is first converted into operational representations to perform score analysis and feature extraction, and then extract state samples (schema-events and -voices). The *memory-data* source (Figure 3, PROTOTYPES) is where schemata prototypes reside and are being recalled for recognition. The *feedback* source (Figure 3, FEEDBACK) inputs non-musical information to the model and is utilized to pass annotations, such as schemata labels for measure ranges. The run-time information of a session is written to a *session log* (Figure 3, RUN-TIME LOG). The learning operations are governed by a system *profile* (Figure 3, PROFILE).

In general, the classifier creates and compares two classes of information: a) *schematic states*, either schema-events or schema-voices, and b) combinations and/or progressions of (a). The three basic operations that connect these information sources all relate to similarity and are: i) recognition, during which states and schemata from memory and data are compared, ii) evaluations, where the results from (i) are compared with annotations, and, iii) adaptation, where the class similarity function of a schema type is updated to include new entries to the example base (the inclusion of a training observation).

The model also employs a number of task-specific functions with smaller scope that aid the similarity functions described above. For example, the results of feature extraction (Figure 3, FE) are sent to both grouping functions (Figure 3, GF, to extract schema-state and progression samples) and a schemata prototype instantiation function (Figure 3, SPi, to create comparable instances from schemata prototypes).

[1] See `https://tinyurl.com/y5x2d99j` for code and data.

3.3 Making music observations

The processing path for each music input element (a complete sonata part) starts with the conversion of the MusicXML file into operational encodings (datapoints and 'minimal segments' [16]) to perform tonal and harmonic score analysis and feature extraction, utilizing formalisms in symbolic music processing from [14] and [4]. Next, stylistic reduction, a combination of the aforementioned analyses for the extraction of schematic information, selects material from the score for the sampling of schema-states (schema-voice and -event feature sets).

In addition to notated information available from the MusicXML encoding, information about metric, tonal, and harmonic properties is extracted utilizing task-specific algorithms. Rhythm is extracted from the notated time signature of the input file and is embedded to operational representations in the form of *metric strength* using [5]. Tonality is extracted from notation using 'key' and 'mode' attributes from MusicXML, but also using probe-tone profiles [12]. Harmony is extracted in segments using the *HarmAn* algorithm [16]. Complex voice separation is not undertaken – rather, the outer notes, based on absolute pitch, are considered to comprise melodic and bass movements.

3.3.1 Sampling schematic-states

After score analysis and feature extraction, the system has enough information to extract schema-state samples. A schema-voice is considered a monophonic sequence of datapoints whose adjacent temporal interval is constrained by extracted features that relate with metric information. The extraction of schema-voices, often termed as music n-grams, is in compliance with voice-leading rules [11]. The schema-event sampling algorithm starts with score analysis information and 'minimal segments', and generates *schema-events*, a type of temporal reduction with similarities to *time-span* reduction described in [13]. First, each 'minimal segment' is assigned positional and (adjacent) transitional 'significance' from two algorithms that weight parameters from the 'minimal segment' properties and harmony. Then, a 'minimal segment' merger creates samples from pairs of 'minimal segment' elements by combining and selecting values for the schema-event characteristics.

The positional 'significance' of each 'minimal segment' is found using (1) and the weights in Table 1:

$$poSig = \tfrac{1}{3}(dpQ + bStr + card) + outVp \qquad (1)$$

where *poSig* is the significance of the 'minimal segment', *dpQ* is the overall quality of datapoints in the 'minimal segment', *bStr* is the beat-strength value, *card* is the cardinality, and *outVp* is the bonus from movement in outer voices.

Transitional 'significance' is calculated in a similar manner but also considering the changes between the adjacent 'minimal segment' elements, mainly in harmony.

A schema-event sample has ontime and duration and all the properties of a prototype schema-event but most importantly, each schema-event sample is rated with the

Feature	Factor
beat strength	2
harmony	2
cardinality	1.5
outer voices	2
complete datapoint in 'minimal segment'	1
starting datapoint in 'minimal segment'	1
ending datapoint in 'minimal segment'	0.25
middle datapoint in 'minimal segment'	0.125

Table 1. The manual weights for *positional* 'significance' (poSig) of single 'minimal segment' elements.

combined 'significance' ratings of its 'minimal segment' elements, allowing further thresholding and selection.

For example, the total number of datapoints and 'minimal segment' elements for the excerpt in Figure 2 is 124 and 95, respectively. Thresholding 'minimal segment' elements using above-average positional 'significance' ratings, event-sampling returns a total of 107 schema-event samples in the following form: ((measure, time signature, beat), (pitch-value of each voice's datapoints: [melody, bass]), (positional 'significance'), (transitional 'significance'), (melody:harmony:bass, in scale-degrees)). For example, the first sample of the excerpt in Figure 2 is:

$$((1, 4/4, 1.0), [C5*, C4], (4.00), (2.00), (1 : - : 1))$$

At a later stage, the type of schemata and the position in which they appear in prototypes are also added to each sample.

With the extraction of schematic material, in the form of schema-voices and schema-events, we have extracted low-level observations, i.e. schema-states. These elements are utilized later for the creation of complete schemata-instances (high-level observations).

3.4 The memory module

The *memory module* stores the schemata prototypes and handles creation, update and instantiation operations of each prototype. It is the knowledge-base, a repository of generic prototypes starting with definitions from [10], where each schema prototype is represented as a sequence of schema-events (please see Figure 1).

3.5 Recognition

A core task of the schemata classifier is to assign/identify prototype schema-states in low-level observations and complete schemata prototypes in high-level observations, with the latter consisting of pairs of schema-voices and progressions of schema-events.

From the data part, post-stylistic analysis (please refer to the data processing path in 3.3), the information of input elements is converted into a set of state samples for voices and events. From the memory part, a target space consisting of schema-states and prototype schemata instances is created according to extracted features (Figure 3, SPi).

3.5.1 Creating and thresholding schemata search space

After the extraction of schema-states, the classifier applies an observation method to select and group state-samples in schema-instances that are comparable to schemata prototypes. Initially, schema-state samples are thresholded based on their 'significance' ratings and, optionally, according to their similarity with prototype states. When forming schemata samples, state-groups are filtered by number of events (minimum/maximum), a minimum/maximum duration, and temporal regularity [6, 18].

3.5.2 Similarity between prototypes and constructs

When comparing extracted and prototype states and schemata, these can either be exact or different, with the latter case being comparable with a similarity metric.

3.5.3 Proto-state similarity

A schema-voice has a single melodic movement and is characterized by the number of notes it contains. Matching a schema-voice sample with a prototype schema-voice is a sequence-similarity task and two voice-states are the same, if their content is the same, regardless of the temporal relations of the datapoints that comprise them. Considering the order of appearance, the position of a datapoint in each voice-state, the difference between schema-voice samples and prototype schema-voices is measured as the sum of differences of same position datapoints, also known as the Hamming distance. For example, the distance d_H between schema-voices (in scale-degrees): a) 1,7,5,3, b) 1,7,4,3, and c) 6,5,4,3, is: $d_H(a, b) = 1$, $d_H(a, c) = 3$, and, $d_H(b, c) = 2$.

The similarity between event-states is hierarchical, considering two layers of similarity for harmonic and melodic information, and Boolean, meaning that differences are not quantified. The harmonic information of two schema-events can differ in type of harmony, expressed within the tonal context (e.g., I, II, etc.) and in the arrangement of the notes within the chord, the type of chord inversions (e.g., 53, 63, etc.). To get a single numeric value from the multi-feature comparisons of schema-events and maintain the types of difference, we consider different decimal powers for each type of difference. Thus, difference in harmonic type equals to 10^3, in chord type, to 10^2, in melody, 10^1, and in bass, 10^0. Therefore, the possible values from a schema-event state comparison are equal to 2^4 (0, 1, 10, 11,. . ., 1110, and 1111). For example, the comparison between schema-events c_{SE} with values (bass, bass-intervals, melody) a) (1, 53, 5), b) (1, 63, 3), and c) (3, 63, 1), is $c_{SE}(a, b) = 1110$, $c_{SE}(a, c) = 111$, and $c_{SE}(b, c) = 1011$. Temporal and metric information is not considered for single event-states comparisons.

After the comparison with prototypical schema-states, each schema-state sample can be tagged with the schema-label and index it appears in prototypes.

3.5.4 Musical schemata similarity

Musical schemata similarity is handled as a sequence similarity problem, similarly to voice-state similarity, but in-

stead of counting the number of pitch-differences in same order datapoints, schema-event differences are counted instead. Thus, similarity between two schemata structures can be expressed as the overall percentage of common schema-events.

Approximation in schemata recognition is similar to the methods presented in [3], using local, γ-, and global, δ- variability thresholds, for schema-events and progressions respectively. Thus, when comparing schema-event progressions, there are two approximation thresholds, one limiting the number and type of differences between schema-events of the same position, and another limiting the number of differences in the complete schema-event progression.

3.5.5 Recognition workflow

The recognition process begins with the comparisons between prototypes and extracted state-samples. First, each schema-state sample is tagged with the id and index of their matching prototypes. Then schema-sampling occurs, creating high-observations – progressions of schema-event samples based on regularity and 'significance' rating thresholds.

3.5.6 Schemata recognition output

The recognition process returns schema-labelled segments of the score and a set of high-observations, rated with the degrees of matching (percentage) with stored prototypes.

3.6 Learning prototypes

Learning musical schemata prototypes is about the extraction and update of schema-event progressions from annotated observations. To achieve this goal, we consider an example-base for each schema prototype class – a repository of observations within a music excerpt. After the recognition process, if the input from feedback suggests a label for a temporal region, then all observations of that segment become training observations and are added to the example-base for that particular schema-label. Maintaining an example-base for each schema-type, we retain access to all the information therein. Processing the elements of the example-base, we examine a class function that extracts relations that are common to all observations that are stored in the example-base repository of a schema. The class function validates all the example-base and is also used for recognition of unlabelled observations.

3.6.1 Class similarity function

To extract prototypes from an example-base of a schema class, we need to identify the harmonic and melodic relations that are common in all exemplars. The algorithm first generates schema-samples and returns variable-length progressions of schema-events that are then converted into harmonic and melodic progressions within the tonal context of the segment (in chord and scale degrees). These contextualized progressions are then sorted by number of schema-events, and the progressions with the highest frequency of appearance and maximum number of events are selected for prototypes.

3.7 Evaluation

The tasks of recognition and learning are evaluated in terms of recall and precision with schemata annotations in measure-level detail. The comparison of the recognition results is Boolean, meaning that the temporal range and label of a recognized schema must match exactly; otherwise they will be considered false-positives.

4. COMPUTATIONAL EXPERIMENTS

We tested four configurations of the musical schemata classifier to gradually achieve the goal of prototype extraction. First, we tested a pattern matching configuration aiming for maximum accuracy in identifying a single schema-type (label I). The second scenario examined approximate matching of a single-schema (label R). The third configuration tests approximate matching of a schema-family, including variations (label F). The last scenario tests the learning algorithm and the extraction of a prototype for annotated examples (label L).

4.1 Maximum accuracy for a user-defined schema

The first model configuration for the schemata classifier examined methods for the extraction of schema-voice states and exact schemata matching for two schemata types, namely 'Meyer' and 'Prinner' (please see Figure 2 for their generic types). The configuration uses datapoints, extracted tonalities, and no harmonic information. The algorithm creates a search-space of schema-voice samples and filters them in pairs, with temporal regularity, to compare them with schemata prototypes. There is no variation in matching and no information preservation after each input element. A recognition example of the algorithm and its configuration is shown in Figure 4.

4.2 Approximate recognition for multiple user-defined schemata targets

The task of approximate recognition of multiple schemata prototypes is examined with two model configurations, matching 'Meyer', 'Prinner', and 'Cadence' schemata. The first case searches for generic approximations (omissions and thresholds in similarity metrics) of each schema family prototype, by first tagging all the extracted schema-states with labels and positions of similar corresponding elements in prototypes and then recognizes complete schemata by combining elements with the same schema class tag under temporal limitations. The second case recognizes approximations of schema families (collections of variants), performing comparisons between schema-samples and class functions that validate all the variants of a schema class. The type of matching approximation of the second case includes those of the first case, but also hierarchical comparisons, considering harmonic similarity first, as a prerequisite, and voice similarity second.

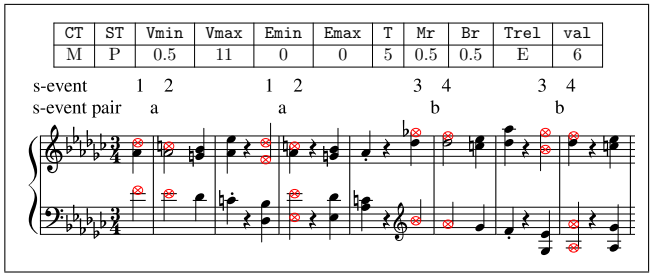

Figure 4. Identification of a generic 'Meyer' instance. The parameters from left to right: Manual search configuration for generic 'Meyer' prototype with temporal distance between events in a range between 0.5 and 11 beats, and 0 for schema-event durations, for a single tonality (5) and 0.5 regularity threshold for temporal intervals in each melodic movement, and maximum (6) thresholding in inter-event temporal regularity. From Ludwig van Beethoven, Piano Sonata in C-sharp minor op.27 no.2 2nd mvt mm.1-7.

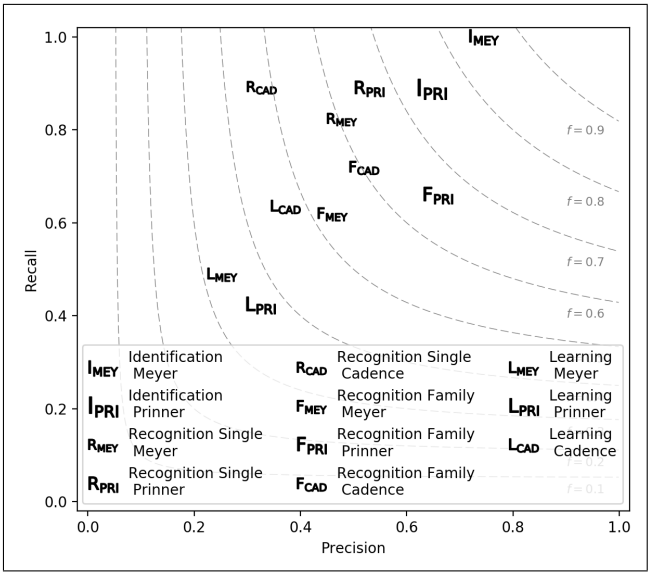

Figure 5. The F_1 score of all musical schemata identification models.

4.3 Approximate recognition of extracted schema family type from examples

This computational experiment examines prototype extraction and classification. The algorithm maintains repositories of exemplars for each schema-type separately, from which prototype forms of schemata are extracted utilizing a class function. The validity of the extracted prototypes is examined by evaluating the recognition performance with the extracted prototypes on previously unseen examples.

5. RESULTS

The identification task (Figure 5, $I_{<SCHEMA>}$) achieved high recall and precision for both schemata types, indicating that basic music information processing of the classifier is working as it should.

The recognition models, due to approximate matching, have increased recall but lower precision (Figure 5, (Figure 5, $R_{<SCHEMA>}$, $F_{<SCHEMA>}$). The first configuration that tests family-prototype generic approximation (Figure 5, $R_{<SCHEMA>}$) can be very imprecise, as approximating the prototypical form can yield completely unrelated patterns. The second configuration (Figure 5, $F_{<SCHEMA>}$) has slightly lower recall from the first but increased precision, due to more targeted approximations with the use of a class function.

The learning model (Figure 5, $L_{<SCHEMA>}$) achieved low recall, due to occasional erroneous prototype extractions and even lower precision, because of the highly-generalised extractions of prototypes.

6. FINDINGS AND DISCUSSION

This paper presented a prototype model architecture for musical schemata classification. It is among a small handful of computational models for extracting instances of such high-level music-theoretic concepts from input staff notation.

The computational model that was developed for a musical schemata classifier was tested under different tasks and configurations, and proves to be a reliable framework that can facilitate classification operations regarding musical patterns. The use of structured music information (i.e. the 'low' and 'high' observations) enabled high-level operations such as comparisons of abstract patterns, and the methods that were developed to produce them can be further fine-tuned. Moreover, the maintenance of an example-base proved very helpful in prototype extraction, as it provides transparency of operations. Furthermore, the repository of a learning session can be reused in other sessions, transferring learned knowledge.

The model was tested on a small dataset and its expansion both in number and types of schemata annotations is a natural future task. There are numerous improvements that can be applied in this prototype model by fine-tuning individual functions. For example, the sampling mechanisms for schema-states and progressions can become adaptive to local context, instead of global, part-wise features, and thus become more accurate. Another interesting and informative development would be the comparison of this model's performance with popular classifiers for sequential data, such as convolutional and recurrent neural networks. In addition, extending the classification capabilities of the model to include information concerning the position and function of schemata within phrases could be useful for both learning and discovery of schemata. Lastly, even the current abilities of the schema classifier can be utilized to support content-based retrieval on digital score libraries. The sampling method is independent from the model and, thus, large databases of digital scores can be represented as samples, or even symbolic fingerprints [1]. In addition, on top of a database representation for music scores, a query language for music patterns (as musical schemata) would provide a flexible interface for a user-defined content-based music information retrieval.

7. ACKNOWLEDGEMENTS

This research has been funded by the Faculty of Computing, Engineering and Media at De Montfort University.

8. REFERENCES

[1] Andreas Arzt, Sebastian Böck, and Gerhard Widmer. Fast identification of piece and score position via symbolic fingerprinting. In *13th International Society for Music Information Retrieval*, pages 433–438, Porto, Portugal, 2012.

[2] Vasili Byros. Meyer's Anvil: Revisiting the Schema Concept. *Music Analysis*, 31(3):273–346, 2012.

[3] Emilios Cambouropoulos, Andreas Katsiavalos, and Costas Tsougras. Idiom-independent harmonic pattern recognition based on a novel chord transition representation. In *3rd International Workshop on Folk Music Analysis*, Amsterdam, Netherlands, June 2013.

[4] Darrell Conklin and Ian H Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, 1995.

[5] Michael Scott Cuthbert and Christopher Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. Utrecht, Netherlands, August 2010.

[6] Katrien Foubert, Tom Collins, and Jos De Backer. Impaired maintenance of interpersonal synchronization in musical improvisations of patients with borderline personality disorder. *Frontiers in Psychology*, 8, 2017.

[7] Robert Gjerdingen. *A classic turn of phrase: Music and the psychology of convention*. University of Pennsylvania Press, 1988.

[8] Robert Gjerdingen. Using connectionist models to explore complex musical patterns. *Computer Music Journal*, pages 67–75, 1989.

[9] Robert Gjerdingen. Categorization of musical patterns by self-organizing neuronlike networks. *Music Perception: An Interdisciplinary Journal*, 7(4):339–369, 1990.

[10] Robert Gjerdingen. *Music in the galant style*. Oxford University Press, 2007.

[11] David Huron. Tone and voice: A derivation of the rules of voice-leading from perceptual principles. *Music Perception: An Interdisciplinary Journal*, 19(1):1–64, 2001.

[12] Carol L Krumhansl. *Cognitive foundations of musical pitch*, volume 17. Oxford University Press, New York, 1990.

[13] Fred Lerdahl and Ray Jackendoff. *A generative theory of tonal music*. MIT press, 1985.

[14] David Lewin. *Generalized musical intervals and transformations*. Oxford University Press, 2007.

[15] Alan Marsden. Schenkerian analysis by computer: A proof of concept. *Journal of New Music Research*, 39(3):269–289, 2010.

[16] Bryan Pardo and William P Birmingham. Algorithms for chordal analysis. *Computer Music Journal*, 26(2):27–49, 2002.

[17] David E Rumelhart. Schemata: The building blocks of cognition. In *Theoretical issues in reading comprehension*, pages 33–58. Lawrence Erlbraum Associates, 1980.

[18] James Symons. Temporal regularity as a key to uncovering statistically significant schemas in an eighteenthcentury corpus. In *annual meeting of the Society for Music Theory, New Orleans, LA, November*, 2012.

[19] Tillman Weyde. Modelling cognitive and analytic musical structures in the musitech framework. In *UCM 2005 5th Conference" Understanding and Creating Music", Caserta*, pages 27–30, 2005.

[20] Tillman Weyde. Automatic semantic annotation of music with harmonic structure. 2007.