VExPat: An Analysis Tool for the Discovery of Musical Patterns

Hugo Santana, Márcio Dahia, Ernesto Lima, Geber Ramalho

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 - CEP 50732-970 – Recife – PE – Brazil

{hps,mlmd,etl,glr}@cin.ufpe.br

Abstract. *This paper describes VExPat, a system that assists on the task of extracting patterns from musical sequences. Although there are known algorithms that perform this task efficiently, none of the existing analysis tools provide this functionality. Furthermore, VExPat provides a user-friendly interface, with score visualization and musical execution facilities, allowing its use even by computer non-specialists, which is the case for a large number of musicologists. Another innovation is the use of MusicXML, an extensible and user-accessible format for representation of music. Finally, it is proposed an architecture for dealing with different internal music representations, and also an object oriented model for a score viewer, both based on MusicXML.*

1. Introduction

Sequential pattern discovery corresponds to the task of, given one or more sequences, finding sets of segments (subsequences) that are significantly similar from some point of view. These sequences may be constituted by any kind of symbolic elements, from DNA or proteins information, widely used on computational biology applications, to note pitches, durations and other kinds of musical abstractions, on the music domain.

Applications for musical sequential pattern extraction[1] range from the most recent research on content-based retrieval of music, where the indexing structures used may be generated using pattern discovery techniques, to pattern oriented music analysis, work that has been done manually for years by musicologists. Some systems for automatic generation of music or musical accompaniment [Ramalho 1999] also deal with this problem.

Although there are a number of existing algorithms that performs this task efficiently [Meredith et al. 2003], there is no analysis tool that provides this functionality. The most they do is pattern recognition (i.e. they can search for a given musical fragment), what is much less than what computer systems may provide to help on this kind of analysis. Also, in order to these algorithmic resources become available to a large number of musicologists, who are typically computer non-specialists, it is necessary to provide a user-friendly interface, with visualization of the music material

[1] The terms pattern discovery, extraction and finding will be used interchangeably to refer to the task of automatically finding repetitions on sequential data. A more precise discussion about the terminology is provided in [Rolland 2001] and [Cambouropoulos 1999].

in a familiar way to them (like scores, for instance), and musical execution of this material.

This paper presents VExPat, a system for computer-assisted musical pattern extraction, and focus on some interesting aspects of its modeling. Besides performing pattern finding, VExPat satisfies some prior requirements [Kornstädt 2001], collected with musicologists, which are usually not followed by analysis systems, such as providing visualization and musical execution of the musical material. To satisfy these requirements, VExPat has to deal with some issues on data representation: How to represent musical information on a system that performs such different tasks? How should end-users fulfill the system with this information? Also, how to obtain an extensible framework of data representation, satisfying different user needs? To deal with these issues, VExPat is assisted by MusicXML [Good 2001], a format for music representation based on XML (extensible mark-up language). Although visualization facilities are not as rich as provided by commercial tools such as Finale[2], a MusicXML-based object oriented structure for score rendition was conceived and partially[3] implemented, giving a simple but functional result, and some insights into this issue.

2. Requirements

This section describes VExPat's requirements. The main non-functional requirement is the usability (to be used by musicologists), which influenced most of the functional requirements following.

The main functional requirement is to provide automatic discovery of patterns on musical sequences. These sequences may be constituted by any kind of musical abstractions (possibly multidimensional, for two or more different abstractions), constituting the internal analysis representation. Another functional requirement is score viewing, allowing users to compare elements visually. VExPat should not only display scores of the analyzed material, but also show the analysis results (patterns found) into these scores. Finally, VExPat should be able to play any piece of the material, and also the analysis results.

A requirement for data representation, taking in account a problem that exists in many analysis tools (see section 3.2), concerns the choice of a format for data acquisition accessible to the users, that is, a format which the user's material is already on it, or which they can easily convert to (or acquire on it). Furthermore, this representation format should be semantically complete enough to allow the system to perform the different functional requirements described, as well as extensible enough, to allow the representation of any new musical abstraction that musicologists may need.

3. State of Art

This section describes related work on pattern discovery algorithms, music representation and analysis tools.

[2] Finale is a trademark of CodaMusic Incorporated - www.codamusic.com

[3] Some MusicXML nodes were modeled but not implemented.

3.1. Pattern Discovery Algorithms

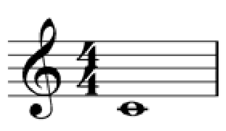
Among many different approaches for discovering musical patterns, all of them have a main process in common: they must compare music segments, and determine if they are similar or not (matching). This section describes some characteristics of SIATEC [Meredith 2001] and FlExPat [Rolland 2001], the two main existing algorithms for pattern discovery, based on how they perform matching and their computational costs.

SIATEC is a geometric approach for pattern discovery. It can deal with polyphonic data, and allows approximate matching. Although SIATEC matching allows gaps and transposition, it is restrictive to other kinds of approximate matching (i.e. the only differences allowed are translations and scales of the elements compared). FlExPat proposes a similarity assessment using the edit-distance paradigm, which allows more flexibility on approximate matching. Also, the edit-distance has already shown to be effective as a similarity measure for music data [Mongeau & Sankoff 1990]. FlExPat is also less computationally expensive, using dynamic programming to achieve $O(N^2)$ complexity (time and space), against $O(N^3)$ for SIATEC.

3.2. Data representation: MusicXML

There are a number of music representation formats available [Selfridge-Field 1997], both commercial and academically. Some of them are well suited for specific purposes only, while others are more flexible. For instance, MIDI is best suited for music sequencing applications, and formats like NIFF or SCORE for score rendering. Although information in these formats is widely available, they are ill suited for analysis. MIDI data focuses on pitches (without enharmonic spelling), while durations are often not quantized, and score notation formats focuses on a layout-oriented information. Analytical formats such as Humdrum, best suited for analysis applications, are more extensible, but as this extensibility is not done on a standardized way (over 40 predefined humdrum formats exists, which can be freely combined), applications cannot count on users having specific representations of its musical material in this format.

We show on this section a format that tries to unify music representation: MusicXML. Based on XML, MusicXML provides a human readable, semantically rich format, yet an extensible representation. MusicXML integration with acquisition tools, such as music OCR systems, score editors, and conversors from and to different formats makes it a user-accessible file format. Figure 1 shows a MusicXML example, and its corresponding score representation.



```
<measure number="1">
    <attributes>
        ...some measure attributes here...
    </attributes>
    <note>
        <pitch>
            <step>C</step>
            <octave>4</octave>
        </pitch>
        <duration>96</duration>
        <type>whole</type>
    </note>
</measure>
```

Figure 1. MusicXML example

Another characteristic of MusicXML is that, as an hierarchical format, it can provide parallel representations of the musical information. For instance, the nodes "duration" and "type", on Figure 1, although having a semantic dependence, they are defined separatedely, allowing the same MusicXML file to represent a precise musical information (as the note was actually played), and its standard notation. With these features, along with the easy to implement XML facilities, MusicXML satisfy the requirements for representation exposed on section 2.

3.3. Related systems: Humdrum and JRing

The widely academically used Humdrum consists on a set of tools (Humdrum toolkit) and an extensible syntax (Humdrum syntax). Between a huge set of analysis capabilities, the Humdrum toolkit provides pattern recognition functions, where users can execute a query for a given musical fragment.

However, Humdrum toolkit does not provide a pattern discovery tool. Another criticism for Humdrum is that building analysis expressions on it is essentially a programming task, keeping away musicians or musicologists, typically computer non-specialists, from effectively using it. The JRing system [Kornstädt 2001] is a front-end to Humdrum, providing score visualization, cataloguing facilities and some already defined humdrum queries. However, some criticism may be done also to JRing: it uses a Humdrum/SCORE file format, which there isn't almost any musical material available, and, as Humdrum, it does not provide automatic pattern discovery.

4. VExPat

This section describes some aspects of VExPat: its user interface, architecture and main modules. Basically, two main project decisions were taken on the development of VExPat: use MusicXML as the end-user file format, and implement FlExPat algorithm for pattern discovery.

4.1. Architecture

Figure 2 shows VExPat architecture, detailing the data flow since a MusicXML file is opened until the musical information reaches its specific module. Each box on Figure 2 represents a VExPat module. The basic data flow consists on the following steps:

1. The XML parser module[4] does the syntactic analysis on a MusicXML file, and outputs an XML syntactic tree;
2. The XML analyzer module consists on a set of specialized components, each one responsible for outputting a particular internal representation: for analysis, score rendering and musical execution. The component for analysis is itself modularized on specific components, each one responsible for extracting a musical abstraction (absolute or relative pitches, durations, onset intervals, etc);
3. The manager module is responsible for maintaining a relation between the three different extracted representations, and forwards the specific representation to its corresponding module (analysis, score renderer or musical execution module).

[4] VExPat XML parser module was implemented using MicrosoftXML DOM parser.

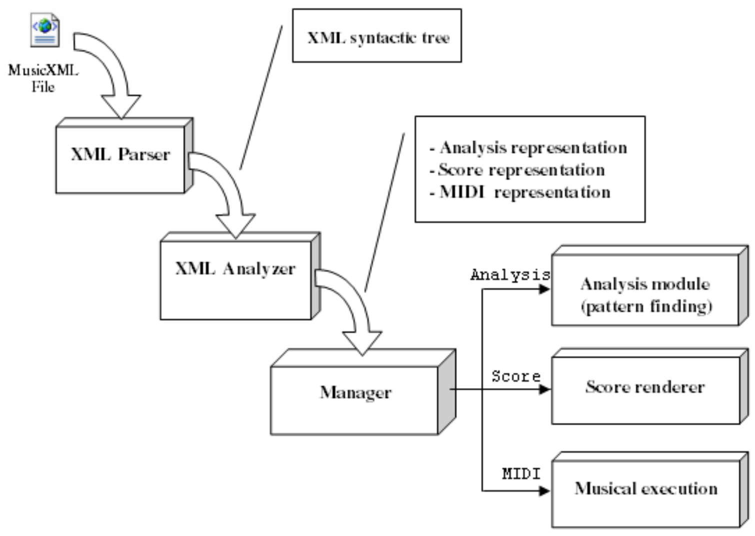


Figure 2. VExPat architecture and main data flow

4.2. User interface

Figure 3 shows VExPat user interface. As shown on the figure, more than one part of a music (or different musics) can be opened at the same time, and patterns found are shown with different colors: red for prototypes (see section 4.4), and green for occurrences. Users can navigate between different patterns, and call an animated play for each occurrence.

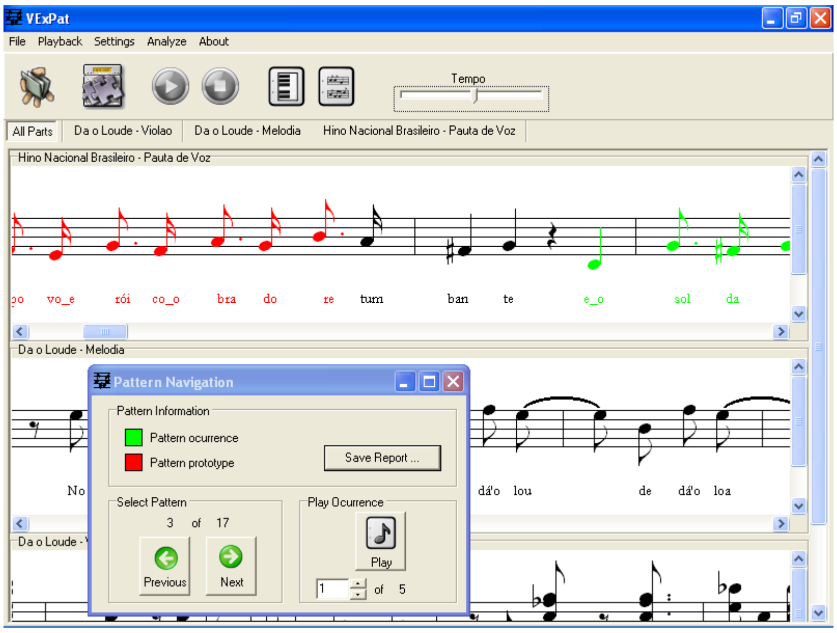


Figure 3. VExPat user interface main screen

4.3. Score viewing module

The Music Notation Toolkit (MNT) [Eales 1999] provides an objected oriented structure for music notation systems. Although some criticism about this model is done on this section, it provides a good insight on the model and ontology of the problem domain.

VExPat took the MNT as an initial model for score viewing, but the following major modifications were done. First, some MusicXML nodes (accidents, ties and others) were not represented as objects by the MNT, breaking the extensibility provided by the format. Also, we found that conceptually these nodes should be objects, as they have individual appearance properties (height, width and coordinates). Second, the Composite design pattern [Gamma 1999] could be used, improving code reuse and allowing an object hierarchy model.

This design pattern is indicated on applications that treat composite and primitive objects as the same, which is the case of score viewers. For instance, a whole note is a primitive object, and a quarter note is a composite (a stem and a notehead), but a client of this object should treat them on the same way. Also, this design pattern allow object coordinates be defined on a relative way (to its component), what is naturally done on a score.

Figure 4 shows the composite design pattern applied on VExPat score viewer module, using the Unified Modeling Language (UML). Follows some explanations about the model [5]:

- GraphicalObjectComposite are composite objects, from high level composite objects, like Score and Staffs, to lower level: measures and notes;
- GraphicalObjectFontBased are primitive objects drawn by musical fonts. Examples are clefs, noteheads, stems, flags, accidents and others.

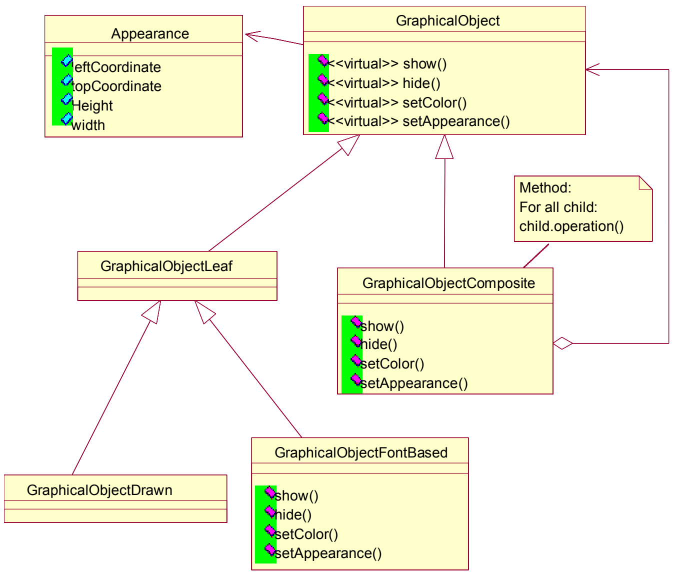


Figure 4. Composite design pattern on VExPat score viewing module

Note that the methods shown have a general implementation for GraphicalObjectComposite and GraphicalObjectFontBased, having to be overriden only

[5] The complete object oriented model is provided in [Santana 2003].

by primitively drawn objects (GraphicalObjectDrawn - ties, beams and other non-font objects).

4.4. Pattern finding module

VExPat shows patterns extensionally: a prototype is shown (i.e. the most representative segment of the pattern), and its occurrences on the analyzed part (or parts). Users can find patterns varying the following parameters: a minimum quorum threshold, a local similarity threshold and weights for musical abstractions.

The quorum threshold is the minimum number of occurrences a subsequence must have to be considered a pattern. The local similarity threshold indicates how similar two subsequences must be to be considered as part of the same pattern. Although FlExPat suggests this parameter as a maximum number of edit-distance operations to transform one of the compared subsequences into the other (i.e. deletion of an element of the sequence)[6], VExPat uses a normalized value, based on the length of the compared sequences.

Users can also give different weights to each abstraction, allowing the discovery of specific kinds of patterns. For instance, if all pitch related abstractions are given a zero weight, and the rhythm related weights are non-zero, the system will find only rhythmic patterns. FlExPat only dealt with note pitches and durations, but other musical abstractions were implemented on VExPat, providing more precise rhythmic patterns extraction. These abstractions are: relative and absolute pitches, absolute durations, onset time since beginning of measure and onset time since beginning of beat.

5. Conclusions

The work presented here brings two main contributions. First, an original analysis tool for pattern extraction was conceived and implemented. Second, the architecture proposed here is innovative, in the sense that there is a lack of references about the use of MusicXML for analysis [Good 2002], and also as a model for tools that, like VExPat, have to deal with different purpose musical representations.

VExPat and its source code are available for download at http://www.cin.ufpe.br/~hps/vexpat/. It was implemented in C++, using Borland VCL[7]. VExPat was shown to musicologists, who gave us an encouraging feedback about the already implemented functionalities, and suggestions for future works. These future works, and some ongoing improvements are:

- Extend the analysis representation (musical abstractions used) to different user needs, creating specific profiles for different styles of music;
- Collect more user feedback. On this issue, VExPat is going to be used on a research project on guitar rhythmic pattern extraction for Brazilian styles;
- Enhance analysis capabilities, possibly integrating it with other analysis tools;

[6] Insertion, Substitution, fragmentation and consolidation are also implemented.

[7] The Visual Component Library (VCL) is a trademark of Borland Incorporated.

- Improve the score renderer module. That can easily be done if open source projects for music notation APIs, such as project XEMO (www.xemo.org), succeeds;

6. References

Cambouropoulos, E., Crawford, T., Iliopoulos, C. (1999) "Pattern Processing in Melodic Sequences: Challenges, Caveats and Prospects" In: Computers and the Humanities, v. 35, p. 9-21.

Eales, A. (1999) "An Object-Oriented Toolkit for Music Notation", M.Sc. Thesis, Rhodes University, Grahamstown.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. and Booch, G., Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1999

Good, M. (2001) "MusicXML for Notation and Analysis", In: The Virtual Score, ed. W. B. Hewlett and E. Selfridge-Field, MIT Press, p. 113-124.

Good, M. (2002) "MusicXML in Practice: Issues in Translation and Analysis", International Conference MAX, p. 47-54

Kornstädt, A. (2001) "The JRing System for Computer-assisted musicological analysis", In: The International Symposium on Music Information Retrieval, Bloomington, IN, USA, p. 93-98.

Meredith, D., Wiggins, G., Lemström, K. (2001) "Pattern induction and matching in polyphonic music and other multidimensional datasets", In: Proceedings of the Fifth World Multiconference on Systemics Cybernetics and Informatics (SCI2001), v. 10, Orlando, FL, p. 61-66.

Meredith, D., Wiggins, G., Lemström, K. (2003) "Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music", Journal of New Music Research (to appear).

Mongeau, M., Sankoff, D. (1990) "Comparisson of musical sequences", Computer and the Humanities, v. 24, p. 161-175.

Selfridge-Field, E. Beyond MIDI: the handbook of musical codes, MIT Press, Cambridge, MA, 1997

Ramalho, G., Rolland, P., Ganascia, J. (1999) "An Artificially Intelligent Jazz Performer", Journal of New Music Research, v. 28

Rolland, P. (2001) "FlExPat: Flexible Extraction of Sequential Patterns", In: The IEEE International Conference on Data Mining, p. 481-488.

Santana, H. (2003) "Uma Ferramenta para Extração e Visualização de Padrões Musicais Seqüenciais", Diploma Project, Universidade Federal de Pernambuco, available at http://www.cin.ufpe.br/~tg/2002-2/hps.zip