SURVEY OF MUSIC ANALYSIS AND VISUALIZATION TOOLS IN PWGL

Mika Kuuskankare
Sibelius Academy
Centre for Music Technology

Mikael Laurson
Sibelius Academy
Centre for Music Technology

ABSTRACT

This paper provides an overview of the current state of music analysis and visualization tools in PWGL. We aim to demonstrate, through several challenging examples, the importance of flexible and extensible music notation and object oriented scripting language. ENP, the music notation tool in PWGL, is used to prepare the analytical examples ranging from contemporary piano pieces to orchestral music. Our scripting language, ENP-script, is used to write analytical rules that automatically display the information in the score. The tight integration between these tools allows us to read the information contained by the score in a flexible and structured way. It is also possible to write new information back in the score or to manipulate the existing data.

1. INTRODUCTION

One of the most interesting applications of PWGL is arguably computer-assisted music analysis. Music analysis as an automated task presents challenges in terms of visualization, representation, and accessibility. Algorithmic control requires a versatile representation of the complex structural and notational information behind the visual front-end of a musical score.

Currently, there are several music analysis applications available (*e.g.*, HumDrum, [1]; athenaCL, [2]) that use text-based representation as a starting point. This kind of approach is usually suitable for analytical questions that are statistical in nature. However, it makes it more difficult to study and verify the result because the analysis is given without the corresponding notational counterpart.

There are some commercial music notation programs that provide a scripting language and/or plug-in interface that allows the user to modify the information in the score. In Sibelius, for example, the user can write scripts by using the built-in scripting language called ManuScript. These tools, however, usually lack flexible and object oriented access to the underlying data. Furthermore, these systems do no provide suitable visualization tools that would support music analytical applications.

The scripting scheme, presented in this paper, can be used to do sophisticated music analysis and visualization. In one hand, the scripting syntax consists of a fairly typical pattern matching scheme with position matching, wild cards, etc. On the other hand, the script is based on Lisp syntax where the operand comes before its arguments. Regardless, script programming in PWGL requires relatively little knowledge of Lisp and a working subset of commands can be learned quite easily, at least by user with background in some other programming languages. Interestingly enough, Lisp dialects are used as scripting languages in a number of applications, with the most well-known being Emacs Lisp in the Emacs editor and Autolisp in AutoCAD. The ENP-script uses the same syntax as its generative counterpart PWGLConstraints [3] and it is not covered in the extent of this paper.

The main software components we use when realizing the examples found in this paper are ENP and ENP-script [4]. The music notation tool ENP, makes it possible to use a set of specialized graphical devices to visualize the analysis information directly in the score. ENP-script, in turn, is used to automatize the analysis process.

In the remainder of this paper we give a number of examples of automated analytical tasks in the context of contemporary music.

2. MUSIC ANALYSIS EXAMPLES

2.1. Harmonic Similarity Measure

The examples in this and the following subsection are realized with the help of a special ENP-expression, called Score-BPF [5]. Score-BPF is a versatile graphical object that can represent break-point functions (piece-wise linear functions) as a part of a musical texture. Score-BPF's can be edited directly in the score or manipulated algorithmically by ENP-script, for example.

Here, we analyze a fragment of a choir piece (Kimmo Kuitunen: Erotessa) using a set class similarity measure called RECREL[6] by Castrén. The RECREL value indicating highest degree of similarity is 0. Two SCs with this value have identical proportioned subset-class contents. The value indicating highest degree of dissimilarity is 100. We aim to display the analysis result as a continuous similarity value function directly in the score.

The first line in our script (Figure 1) demonstrates the object oriented dimension of ENP-script. The keyword *:harmony* allows us to access the harmonic information in the score as a complex object. Because our analysis is dependent of the relationship of two consecutive harmonies, two variables (*?1* and *?2*) are used. The last entry in the first line is used to check that the script is executed only when all the members of the latter harmony are known. [1]

[1] This is due to the internal organization of the scripting engine that

Lines (b-c) access the break-point function that can be seen in the score above the staff.[2] Next, the RECREL value between two consecutive harmonies is calculated (d-e). Lines (g-k) do some assignments and rudimentary checks about the integrity of the analysis data. Finally, in the last row, the RECREL value is inserted in the global break-point function as a new point. The timing information, that is needed to insert the new point in the correct position in the score, is read from the latter of the two harmony objects.

```
a  (* ?1 ?2 :harmony (m ?2 :complete? t)
b    (?if (let ((bpf (nth 0 (break-point-functions
c                             (pwgl-value :bpf))))
d               (s1 (set-class-name (m ?1)))
e               (s2 (set-class-name (m ?2)))
f               (r -1))
g           (setq r
h                 (if (and (> (sc-info 'card s1) 1)
i                          (> (sc-info 'card s2) 1))
j                     (rec-rel s1 s2)
k                   -1))
l           (insert-point bpf (start-time ?2) r))))
```

Figure 1. ENP-script that analyzes and visualizes the RECREL relation between two consecutive harmonies.

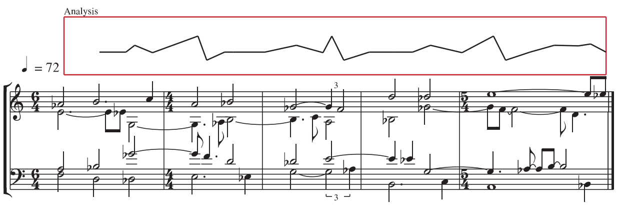

Figure 2. A set class similarity measure analysis shown in the score with the help of a Score-BPF (Kimmo Kuitunen: Erotessa; piano reduction).

2.2. Texture Analysis

In our next example we use an excerpt of a piano piece called *Twine* by the Finnish composer Magnus Lindberg. This time we analyze the thickness of the musical texture.

Unlike in the previous script, we are now using a single harmony object (see Figure 3). We also test if the harmony is complete (a) to ensure that both the highest and lowest pitches of the current harmony are present. In order to represent an area with the Score-BPF, two synchronized break-point functions (b-e) are used. Finally, two points are added in (f-i). The object oriented syntax of the scripting language allows us to easily access to the maximum and minimum pitch value in the harmony, as can be seen in (g) and (i).

2.3. Structural Analysis

We use Twine again as a basis for our next analysis assignment. It is a well known fact that Lindberg makes use of a

```
a  (* ?1 :harmony (m ?1 :complete? t)
b    (?if (let ((bpf-high
c                (nth 0 (break-point-functions (pwgl-value :bpf))))
d               (bpf-low
e                (nth 1 (break-point-functions (pwgl-value :bpf)))))
f           (insert-point bpf-high (start-time ?1)
g                         (m ?1 :data-access :max))
h           (insert-point bpf-low (start-time ?1)
i                         (m ?1 :data-access :min)))))
```

Figure 3. ENP-Script analyzing the textural density of a complex piano piece (Magnus Lindberg: Twine).

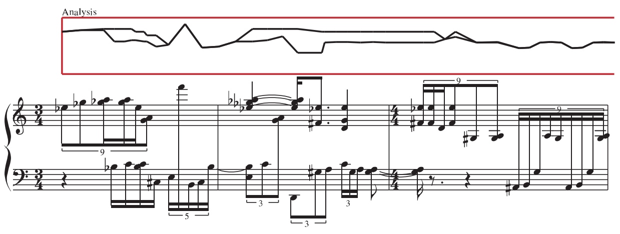

Figure 4. Texture density analysis of Twine by Magnus Lindberg shown as a graph above the staff.

chaconne-type harmonic structure where the progression of the piece is based on a repeated chain of chords. The elements are constantly repeated so that a new cycle starts immediately after the end of the previous one.

We have prepared an ENP-script to analyze this structural dimension. This particular case utilizes the concept of a 'score-sort' (see Figure 5, line a) that is a way to arrange all the notes in a score in a sequence so that each new pitch appears according to a simple set of rules: (1) notes are sorted according to their start-time starting from the notes from the lowest part up, and (2) when there are more than one note with exactly the same start-time the longer notes come first.

The script shown in Figure 5 works as follows. We have two external variables, *:index* and *:chords* (c). The variable :chords contains the chaconne sequence of Twine and :index, in turn, is placeholder for an index value that is incremented every time a new occurrence of a chaconne chord is found. The heart of the script can be found in the lines (g-n). Here, we use a special design in ENP-script, called *:l-filter*, that allows us to go backwards in the chain of notes as long as the given conditions are met. When the end condition is true l-filter returns a list of all the notes that fulfill the given criteria. This list is stored in a variable called *chaconne-notes* (f). Finally, when the notes are found we insert a box-shaped expression (o) in the score and increment the chaconne chord index by 1 (p) thus moving to the next chord.

2.4. Large-scale Example

As our last example we give a complex analysis assignment that uses as a starting point the piece called *GranDuo for 24 wind instruments*, by Magnus Lindberg (see Figure 8 in the Appendix, measures 46–48). The analysis assignment is taken from [7].

The task for our analysis script is to identify pitches that belong to three properties, described below, and mark them in the score using distinguishable shapes.

allows us also to access partial harmonies, i.e., the harmony object is built from bottom up, one note at the time.

[2] All the script examples access special variables through a function called 'pwgl-value' . The variables are created and initialized outside the script. The mechanisms pertaining this process are not covered in this paper

```
a (* ?1 :score-sort
b    (?if
c     (let ((chord (nth (pwgl-value :index) (pwgl-value :chords)))
d           (chaconne-notes))
e       (unless (member (m ?1) chord)
f         (when (setq chaconne-notes
g                     (m ?1
h                        :object T
i                        :L T
j                        :L-filter #'(lambda(x)
k                                      (unless (eq x ?1)
l                                        (if (member (m x) chord)
m                                          T
n                                          :exit)))))
o           (add-expression 'score-expression chaconne-notes))
p         (pwgl-value :index :write (1+ (pwgl-value :index))))))
```

Figure 5. Script making an analysis of chaconne-type harmonic structures.

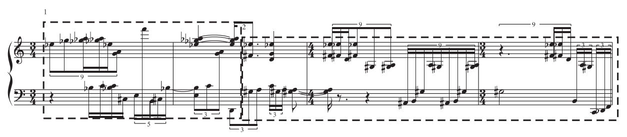

Figure 6. The boxes drawn with a dashed line display the chaconne-type harmonic structure of the piece according to the principles of Magnus Lindberg.

The first element to analyze is a particular ordering of all 12 pitch-classes of the chromatic scale into an arrangement that is called a *wide scale* (Castén, see Figure 9a).

The second element is called the *modified harmonic series* (Ibid, see Figure 9b). A modified series consists of real harmonic series with the exact pitch of each partial rounded to the nearest tempered pitch. Octaves and their multiples are omitted so that each pitch class is represented by the lowest representative in the series.

Thirdly, we mark cases that do not fall either of the aforementioned categories.

The script given in Figure 7 is quite straightforward. It consists of three components. In lines (b-e) we test if the current pitch is a member of the modified harmonic series and not a member of the wide scale (marked with a circle). In (f-i) the roles are reversed and this case is marked with a square. Finally, in (j-o), we test and mark notes that do not belong to either of the properties. These are marked with a small question mark (*?*) that is placed above the corresponding note. Thus the unmarked cases belong both to the modified harmonic series and the wide scale. The result can be seen in Figure 8.

```
a (* ?1
b    (?if (cond ((and (member (m ?1) (pwgl-value :harm))
c                     (not (member (m ?1) (pwgl-value :ws))))
d                (add-expression 'score-expression ?1
e                                :radius 1.0 :kind :circled))
f               ((and (member (m ?1) (pwgl-value :ws))
g                     (not (member (m ?1) (pwgl-value :harm))))
h                (add-expression 'score-expression ?1
i                                :radius 1.0 :kind :box))
j               ((and (not (member (m ?1) (pwgl-value :ws)))
k                     (not (member (m ?1) (pwgl-value :harm))))
l                (add-expression 'plain-text ?1
m                                :expression-print-level
n                                :above-chord
o                                :info "?")))))
```

Figure 7. Script analyzing multiple aspects of pitch organization.

3. CONCLUSIONS

This paper presents some real-life music analytical tasks automated using the built-in music scripting language of PWGL called ENP-script. The musical excerpts are realized using ENP, the music notation package inside PWGL.

We believe that in terms of visualization, representation, and accessibility our system is quite unique: Firstly, PWGL provides an integrated frame-work for music analytical applications. Secondly, ENP provides us with a versatile structural and visual representation of musical information, e.g., the analysis results can be visualized directly in the score using ENP-expressions. Thirdly, ENP-Script offers flexible access to the information contained by the score through an efficient and expressive pattern-matching syntax. Furthermore, All the core components are tightly integrated allowing us to share, modify and represent information in many different ways.

4. ACKNOWLEDGMENTS

The work of Mika Kuuskankare and Mikael Laurson have been supported by the Academy of Finland (SA 114116 and SA 105557).

5. REFERENCES

[1] Huron, D., “Music information processing using the humdrum toolkit: Concepts, examples, and lessons”, *Computer Music Journal*, vol. 26, no. 2, pp. 15–30, 2002.

[2] Ariza, C., *An Open Design for Computer-Aided Algorithmic Music Composition*. PhD thesis, New York University, New York, NY, USA, 2005.

[3] Laurson, M., *PATCHWORK: A Visual Programming Language and some Musical Applications*. Studia musica no.6, doctoral dissertation, Sibelius Academy, Helsinki, 1996.

[4] Kuuskankare, M., *The Expressive Notation Package*. PhD thesis, Sibelius Academy, DocMus – Department of Doctoral Studies in Musical Performance and Research, 2006.

[5] Kuuskankare, M. and M. Laurson, “ENP-Expressions, Score-BPF as a Case Study”, *Proceedings of International Computer Music Conference*, (Singapore), pp. 103–106, 2003.

[6] Castrén, M., *RECREL: A Similarity Measure for Set-Classes*. PhD thesis, Sibelius Academy, Helsinki, Finland, 1994.

[7] Castrén, M., “Aspects of pitch organization in magnus lindberg’s granduo for 24 wind instruments”, *Fourth International Conference on Music Theory A Composition as a Problem*, vol. I, (Tallinn), pp. 61–74, April 3-5 2004.

Appendix

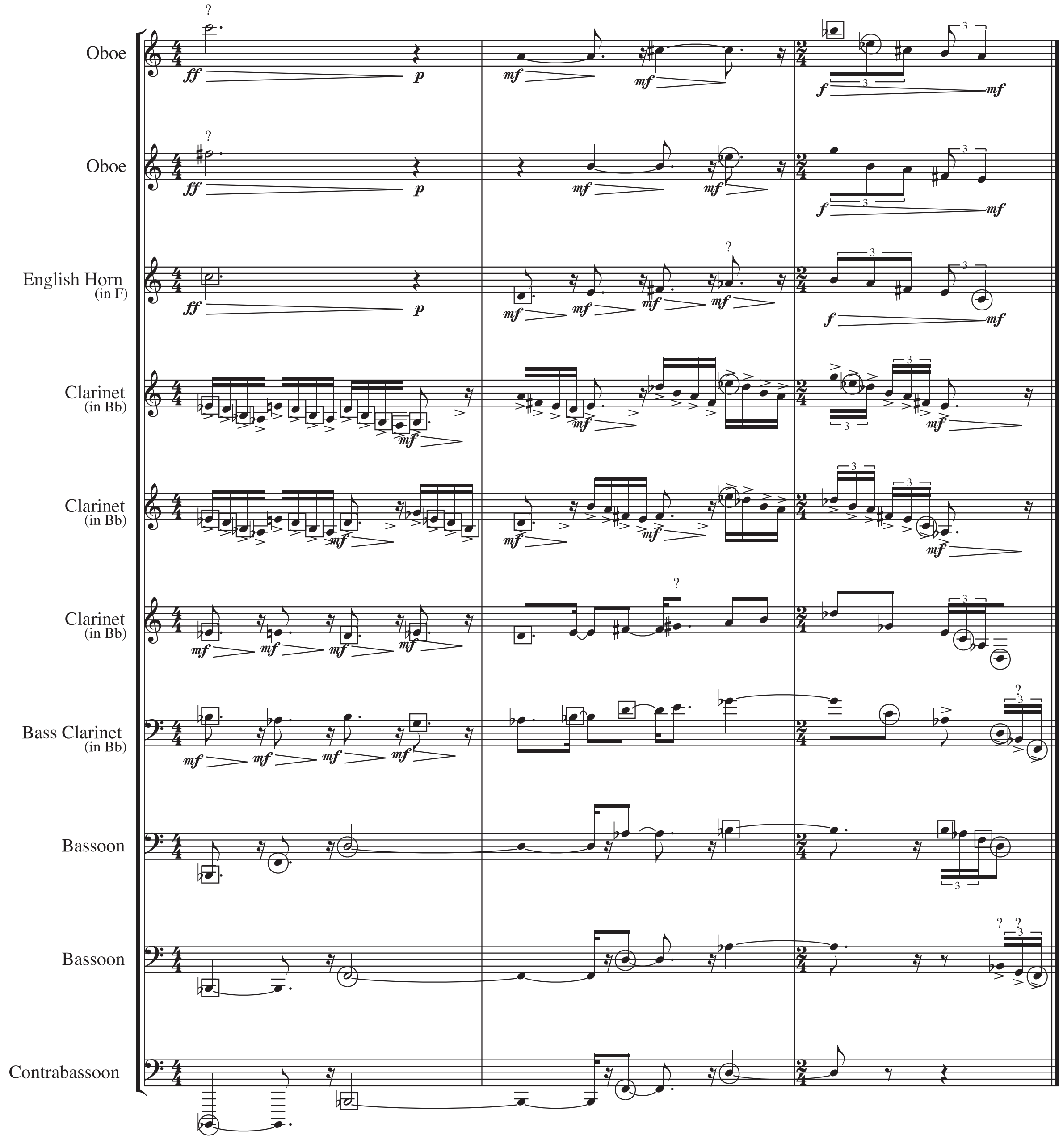

Figure 8. Magnus Lindberg: GranDuo for 24 wind instruments (measures 46–48). Notes that are members of the modified harmonic series but not members of the wide scale are marked with a circle. The opposite case is marked with a square. Notes that do not belong to either of the properties are marked with a small question mark. The corresponding modified harmonic series and the wide scale can be found in Figure 9.

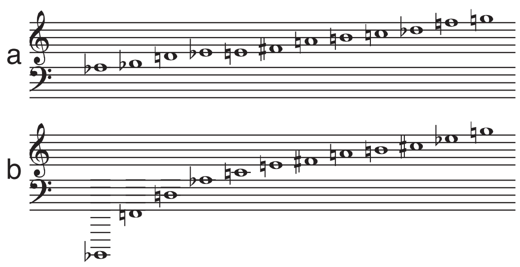

Figure 9. The wide scale (top) and the modified harmonic series (bottom).