Music Segmentation: an XML-Oriented Approach

Goffredo Haus, Luca A. Ludovico

LIM-DICO University of Milan
Via Comelico, 39
20135 Milano, ITALY
haus@dico.unimi.it, luca.ludovico@dsi.unimi.it

Abstract. In this paper, two related subjects are discussed: musical segmentation and the representation of its results in a particular XML encoding, namely MX. About segmentation, there is a comprehensive discussion of the main ideas and the employed operators to implement it. Besides, in this paper we describe the principles of our music XML format.

Introduction

This paper represents the results recently obtained at LIM (Musical Informatics Laboratory, State University of Milan) in the area of musical computer science. In particular, the document is focused on the automatic analysis of musical works in order to recognize and extract musical objects. Our purpose is to encode both music information and corresponding meta-data in a unique data structure. From this point of view, XML provides an effective way to represent musical information at different levels of abstraction. In the format we propose, namely MX, it is possible to correlate notational symbols as well as audio fragments, and printed scores as well as performance files. Our encoding format is particularly suitable to represent information coming from a manual or automatic segmentation process. Thanks to its multi-layer structure, themes and other musical objects can be referred not only to organized symbols in score, but also to audio performances and printed documents. XML encoding and segmentation will be the main subjects of the following discussion.

XML and Music Information

XML provides indeed an innovative way to represent information. The purpose of our efforts is to benefit by XML in order to generate a complete description of musical communication. From this point of view, XML presents a number of remarkable features:

? **Intelligibility:** compared to other encoding formats, a markup language is more intelligible. Thanks to the tag mechanism, the receiver of music communication (both a computer user and a musician) can easily retrieve information. A specific software application is not required, as information is encoded in plain text format.

- **Implementability:** musical applications using XML are implementable. A number of XML-based software were developed, both to represent and to manipulate music information. Possible manipulations include not only simple editing but also meta-data extraction and segmentation.
- **Extensibility:** an XML format can be extended to represent data, meta-data and layers previously ignored. Dealing with a dynamic field such as music production, we can not neglect possible future evolutions: for example, new notational symbols in score or new levels of abstraction in multimedia communication.
- **Hierarchical structure:** music information is strongly structured. A music piece is made of one or more pages, each one containing one or more staff groups, each one containing one or more staves and so on… (parts on a staff, voices in a part, beats for a voice, chords in a beat, notes in a chord). This example shows a hierarchical structure, which can be reflected and properly represented by XML.

Thus, XML is an effective way to describe music information. Nowadays, there is a number of good dialects to encode music by means of XML: MusicXML, MusiXML, MusiCat, MDL etc. (see [11] for a thorough discussion). In particular, we have at least two good reasons to mention MusicXML. First, it can be considered a good and comprehensive way to represent symbolic information. As a consequence, MusicXML was integrated in a number of commercial programs. Among them, it's worthwhile to cite one of the leading applications for music notation: Coda Music *Finale*. And another advantage of MusicXML is represented just by its popularity in the field of music software.

We developed a new XML-based format, called MX. Our approach is different from the aforementioned ones thanks to the following key features: the *multi-layer structure* for music information and the concept of *space-time construct*.

In our opinion, musical information can be (and should be) structured by using a layer subdivision model, as shown in Fig. 1. Each layer is specific to a different degree of abstraction in music information. In our proposal for a common and exhaustive format, we distinguish among General, Structural, Music Logic, Graphic/Notational, Performance and Audio layers (see Figure 1a). For example, MusicXML could be integrated in the more comprehensive MX encoding to implement the Logical Organized Symbols layer, that is symbolic information in score (notes, rests, articulations,…); whereas other common file types can be linked to represent other layers: TIFF for the Notational layer, MP3 and WAV for the Audio layer and so on. There is a good coverage of the matter in [11].

Considering music structure as multi-layered, we need a sort of glue to keep together the heterogeneous contributions. Accordingly, we introduced the concept of spine. *Spine* is a structure that relates time and spatial information (see Figure 2), where measurement units are expressed in relative format. Through such a mapping, it is possible to fix some point in a layer instance (e.g. Notational) and search the corresponding point in another one (e.g. Performance or Audio). Later in this document, a complete example is provided.

Currently, MX is undergoing the IEEE standardization process, as described in [1]: Recommended Practice for the "Definition of a Commonly Acceptable Musical Application Using the XML Language" (IEEE SA 1599, PAR approval date 09/27/2001).

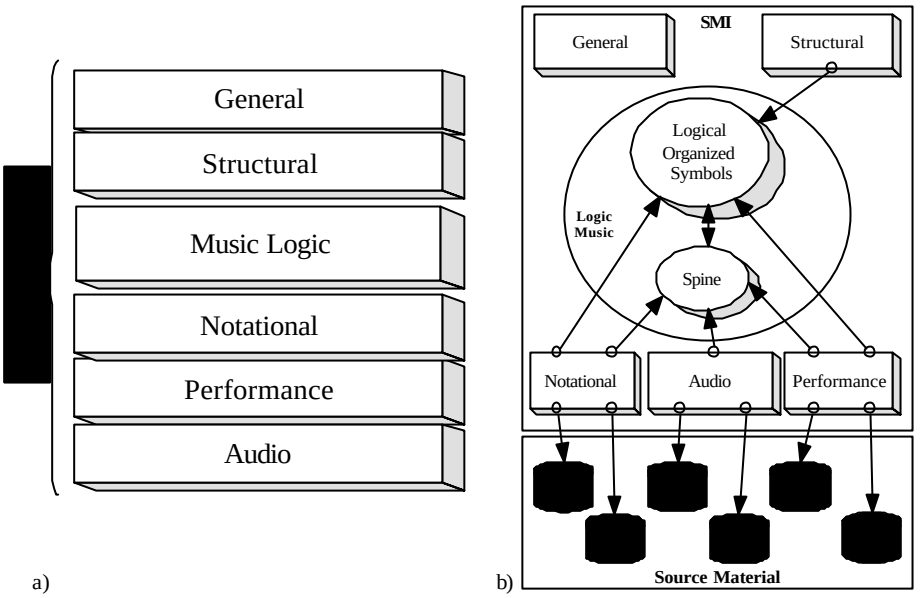

Fig. 1 – (a) Music information layers and (b) relations among them

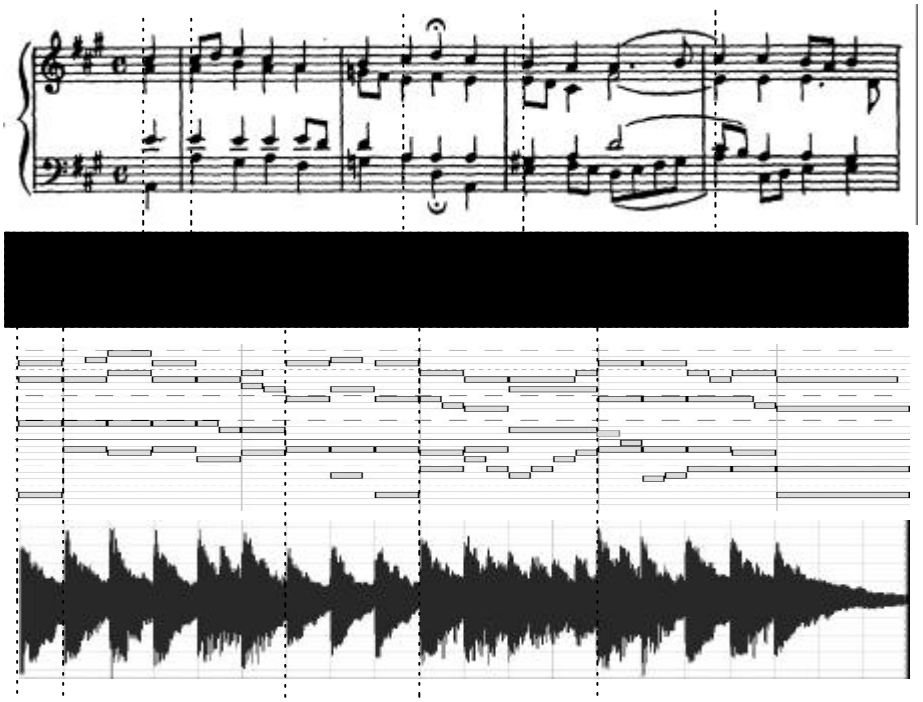

Fig. 2 – Spine: relationships between Notational and Performance/Audio layer

Segmentation

Segmentation implies recognizing and extracting main musical contents. This evaluation can be performed involving either a score or an audio recording. The two approaches are usually referred to as *notational* and *audio segmentation*, and are wrongly considered independent. Studies on relationships between both representations are currently in progress: the purpose is to reduce the distance between the former and the latter, proposing integrated-analysis techniques on audio and notational layer.

The target of segmentation is, at first, the aggregation of significant groups of elements: melody, rhythm or harmony patterns, eventually combined. The next step is the analysis and extraction of music objects, i.e. those parts of a piece characterized by common features. The meaning of the locution "common features" is too general, but soon it will become clear. As mentioned before, musical language is very rich and complex, and information can be conveyed at different levels of abstraction. So, an example of segmentation at notational level could be: "Let's consider all the notes carrying a staccato articulation symbol". This could represent a form of segmentation, but really poor from a musicological point of view. Another type of interrogation could be the following: "Let's take into consideration all the sounds emitted by horns in an orchestra score". Now segmentation could occur either at audio or at notational level, and would have its own musical meaning, indeed. But in the former case it would be a very hard task to extract the horn part from an orchestral tessitura;[1] on the contrary, considering notation, the problem would become even trivial. The question is: among all the possible contents we can search for in music, what is really relevant? What do we expect to determine?

Our approach is trying to partition music in order to recognize themes and other important recurrent musical objects. Our search is oriented toward generative material extraction from scores, including notation, phrasing and, at last, semantics. And we would like to reach this goal by the automatic process of segmentation we will briefly introduce in the following sections.

This computer-based process requires to pay a great attention to the correct evaluation of results. For instance, a strongly repetitive sequence of notes (pitches and rhythmic values) could highlight an important musical figure, such as a theme, as well as an irrelevant accompaniment.

In the structure depicted in Figure 1b, the starting point of our computations is represented by score or performance file formats, such as NIFF, MIDI, and MX itself. These files are parsed by our application software, namely Theme Finder, and transformed into a common notational encoding. Then, analysis process can begin and the corresponding results are finally written into an MX file. In the next section, our segmentation process and employed techniques will be detailed.

[1] This is the problem usually referred to as *demixing*. Many studied about this matter are currently in progress, also at LIM.

The Concept of Musical Object

We can refer to a "musical object" as every type of structured information in any musical language. As mentioned before, our purpose is the extraction of thematic elements. Themes and musical objects are formalizations of general linguistic concepts, thus the risk is to introduce sensitive restrictions based on aesthetic, historical and formal criteria.

Even at basic symbolic level, musical language has a multidimensional syntax, where an interweaving of rhythm, melody and harmony can be found. The musical alphabet is composed by a combination of at least two data for each symbol. The most atomic sound element, called the *note*, joins a sound pitch (frequency) to a rhythmic value (time duration). Usually, musical alphabet is more complicated: for instance, at rhythmic level not only durations but also accents play an important role. And, in order to have a more complete characterization of elementary information, we should also consider the loudness of note sound, and the timbre of the playing instrument .

However, a simplification of the model by considering only rhythm, melody and harmony dimensions can be sufficient. In fact, mental interpretation process doesn't force us to distinguish different instruments (even if it actually does): for instance, music themes are recognized even if they are split on different voices. And loudness is never related to a single language particle but to more complex expressive structures .

Each melodic line, or real voice, is made of notes and is decomposable in two dimensions immediately perceived by the listener. These dimensions are the rhythmical-melodic character (horizontal dimension) and the harmonic character (vertical dimension). The latter aspect is not referred to a single melody, but has to be evaluated together with other contemporaneous melodic lines. This originates the almost independent dimension of tonal functions.

The dimensional complexity just described, even if reduced and simplified, makes musical language harder to be formalized than other natural languages. Besides, only a few elementary harmonic patterns (such as tonic-subdominant-dominant-tonic), some melodic movements (e.g. sensible-root), and some rhythmical punctuation features (pauses, long-duration notes,…) apparently keep a unique definition or a commonly accepted meaning in all historical ages and cultures.

The main problem we have to deal with is the fact that a musical object is recognized also in presence of a slightly different information flow. Variation (in the form of embellishment, transposition, retrogradation, and so on…) is probably the most important method used by composers to develop music contents, and it can not be ignored. Thus, our pattern matching techniques not only will take into consideration literal repetitions, but also a number of conceivable variations and adjustments.

Of course, automatic segmentation is a difficult task to perform. At the moment, human intervention, even if not required, is still desirable. The supposed music themes must undergo a hand-made musicological evaluation, finalized to recognize their expected relevance and their completeness. An automatic process could extract a

musical theme which is too long, or too short, or simply insignificant. That's why a human feedback is still required in order to obtain high-quality results.

Introduction to Musical Operators

Now it should be clear that our goal is the automatic recognition of musical objects, finalized to their representation in a commonly accepted standard format.

In order to find out musical objects through a computer system, a number of musical operators must be defined. Musical operators are an algorithmic tool-set able to identify and report various types of musical objects.

Experience shows that limiting analysis to a single layer would produce a poorly meaningful segmentation. That's why a great effort was done to design and implement data structures and algorithms suitable to multi-layer analysis. In particular, the newest version of our software, Theme Finder 3.0, is able to recognize and automatically extract musical objects using overall information in an integrated way.

As mentioned before, we consider essentially three layers: melody, rhythm and harmony. To each layer corresponds a subset of specific operators. The melodic, rhythmic, and harmonic operators we conceived are able to produce compatible and complementary results, even if they are very different in meaning and operating fashion.

Keeping the computational complexity fair should be one of the purpose of any implementation. To achieve this goal, our integrated analysis employs a "nearly diagonal" approach: computations on different layers occur in different moments, but the overall outcome is produced using all available information. Besides, during the analysis process, the extracted information "flows" among the levels, thus working as a guide for the following musical operators.

Unfortunately, in this short paper it is impossible to list all the operators with their algorithms, functions and structures. [7] and [12] cover this topic in detail. Here we will describe only their general principles and their main functionalities.

Redundancies and variated repetitions are recognized by applying a pattern matching technique. In particular, most operators consider two given sequences of notes, each one containing n symbols, and calculate a vector for each sequence. Of course, the contents of the vector depend on the type of operator being applied. For instance, a melodic operator is interested in pitch evaluation; thus, the vector elements will contain numbers that mathematically describe the pitch information. Then, the contents of the vectors are compared, and once again the way of matching is due to the specific operator. As a result, an error vector is generated. In a certain sense, the error vector measures the distance between two candidate musical objects. If all its components are null, then there is a perfect matching (according to that particular operator); if not, we must introduce a less rigid metrics than a simple binary "matched/unmatched" logic. To evaluate properly varied recurrences, very common phenomena in music compositions, a concept of tolerance must be introduced in the process of analysis.

Rhythmic/structural Operators

This first class of operators focuses on rhythmic analysis. As a matter of fact, rhythmic patterns can be considered a way to describe musical objects.

When talking about rhythm, usually we think about durations. Of course, note (and rest) lengths represent an important aspect of rhythm. As a consequence, one of the key features of our operator is the length matching.

But there is another aspect we must deal with: the accent arrangement. The matching between two rhythmic objects cannot ignore also the congruence of accents. An accent is a "conceptual reference" of a perceptive feature; it can explain the "rhythmic tension" a note assumes due to its position in bar.

Notes can be grouped into equivalence classes, depending on note position in bar, beat (strong time, stasis, pulse) and upbeat (weak time, impulse) of bar, regular or irregular groups, syncopations and so on… Starting from score information, it is possible to use a simple automaton to assign rhythmic accents to each note in a melody. A possible implementation is discussed in [7].

Before performing an automatic comparison of fragments, our operator weights how much the position in bar is discriminating for the algorithm.

The parsing of rhythmic dimension has the advantage of being performed in a linear time cost complexity. Furthermore, the mathematical model is isomorphic to relative musical theory, because music division is based on mathematical rules. Thus, we can affirm that the theory is consistent.

In conclusion, two fragments will be considered completely equivalent from a rhythmic point of view only if both duration and accent equivalence is found. In our implementation, these two comparisons are unified under a single rhythmic/structural operator.

Melodic Operators

Also melody can be investigated by using pattern matching techniques. In this case, variations and their relationships with the original fragment have a particular importance. The way musical sequences are revised ranges from a simple reintroduction of the original phrase in a new tonality to complex counterpoint devices.

Thanks to the melodic operators we introduced, we are able to recognize musical modifications of four types:

? Transposition (real and tonal)
? Inversion (real and tonal)
? Retrogradation
? any combination of the preceding operators.

Transposition implies moving a note or collection of notes up or down in pitch by a constant interval (real transposition) or by a constant number of grades (tonal transposition). The *inversion* of a given melody is the melody turned upside-down. For instance, where the original melody has a rising third, the inverted melody presents a falling third; once again, intervals can undergo a real inversion (e.g., a rising major third becomes a falling major third) or a tonal one (e.g., instead of moving two grades up, melody falls two grades down). *Retrogradation* is the

up, melody falls two grades down). *Retrogradation* is the movement in the opposite sense: the last note of the original sequence becomes the first note in the varied one, and so on…

In order to apply melodic operators, a formal model of musical scale must be provided. As mentioned before, there are two distinct ways to measure pitches, and therefore to evaluate distance between notes:

? a method based on chromatic distance, i.e. on half-tones; this technique is insensitive to the tonal context.
? another based on diatonic distance, i.e. on grades of a scale; this method considers the actual tonality.

In order to implement both real and tonal operators, we need both the enumerative systems: one to calculate the real distance, another to calculate the tonal distance (see [4] and [12] for a detailed discussion).

Melody is a sequence of pitches, comparison is performed on this succession. Let's notice that there is a perfect informative equivalence between a description of absolute pitches and a sequence of melodic interval, where only the first note is eventually described by its absolute pitch whereas the following are identified by the distance from the preceding one. The latter represents a differential approach, the one we choose in our implementation for the following reasons:

1. Human ear naturally evaluates pitch variation between notes instead of absolute frequency of each note. Most listeners recognize a melodic pattern by matching pitch sequential differences (as well as rhythm, of course).
2. In our implementation, absolute values are not important to determine recurrences and variations. This approach allows to code only differential information, saving hardware and computational resources: for instance, a *n*-notes sequence can be represented by a (*n*-1)-items vector.

Harmonic Operators

Harmonic approach completed our analysis system at semantic-contextual level. Melodic and rhythmic/structural operators highlight good local solutions, but they can not provide a structural cognition of the musical meta-language ruling phraseological generation. The aforementioned operators can be used for a syntactic analysis of musical language, but harmony evaluation is a powerful instrument to identify sections and phrases within the composition plan. In this sense, harmony analysis is a helpful cognitive tool for understanding phrase construction and for limiting its scope. Compared to melodic and rhythmic/structural operators, harmonic operators work at a higher level of abstraction, while discovering a deeper structure.

The following operators work serially in order to perform the desired analysis.

The first harmonic operator is *Verticalization*, used to identify the temporal occurrence of a sound event. It is essentially a "selective" function, with a domain defined by all the notes in the score. This function groups notes in chords. Verticalization operator is able to distinguish and manage redundant sounds that harmonic analysis should not consider.

The second harmonic operator is called *Tonal Function Recognition*. This operator evaluates the chord passed as argument, and return a symbol of musical meta-language [17]. Its purpose is not only of recognizing the grade of the scale where the chord is built, but also of providing a musical explanation of the chord function. The final goal of TFR operator is the reduction of a whole set of notes (corresponding to single chords) in a symbol belonging to one of three classes: Tonic, Dominant or Subdominant. These classes correspond to the key harmonic regions in a musical piece. A further symbol of indefiniteness is provided, but it should remain highly improbable.

The following harmonic operator is referred to as *Harmonic Cadence*. Its goal is reducing the hard work of locating musical cadences to a series of simple logical formula. This operator assigns Boolean values to each possible chord sequence, in order to determine which chords may belong to a harmonic cadence. Thanks to TFR operator, now we can deal with every chord (that is a set of notes) as a single symbol in a very limited alphabet, so the task of Harmonic Cadence operator is simpler.

Finally, the *Harmonic Redundancy* operator provides an easy way to collapse redundant information.

Segmentation and MX Encoding: a Case Study

In order to show a practical result of our studies and implementations, a short example of segmentation and MX encoding is now provided.

The piece taken into consideration is a two-voices composition by J. S. Bach: Invention #4 BWV775. Due to the lack of space, only a short part of the whole piece can be shown.

In this example, we report the MX encoding of musical contents, together with a number of other related representations. In particular, notational data can be retrieved from MX, whereas the graphical aspect of score is visible in TIFF format, its performance information is provided by MIDI layout, and the audio layer is represented by the corresponding waveform.

The example is made of two parts. In the former we want to underline the first key aspect of MX code: the “glue” function among different levels provided by the spine layer. This layer is included inside `<logic>` tags. The latter shows the effects of a possible segmentation and the corresponding MX representation. Within MX encoding, this part is marked by `<structural>` tags.

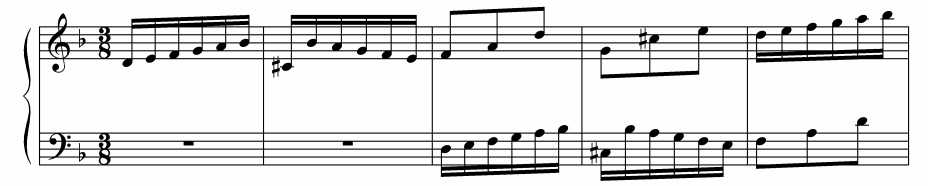

Fig. 3 – Notational layer: TIFF representation for Invention #4

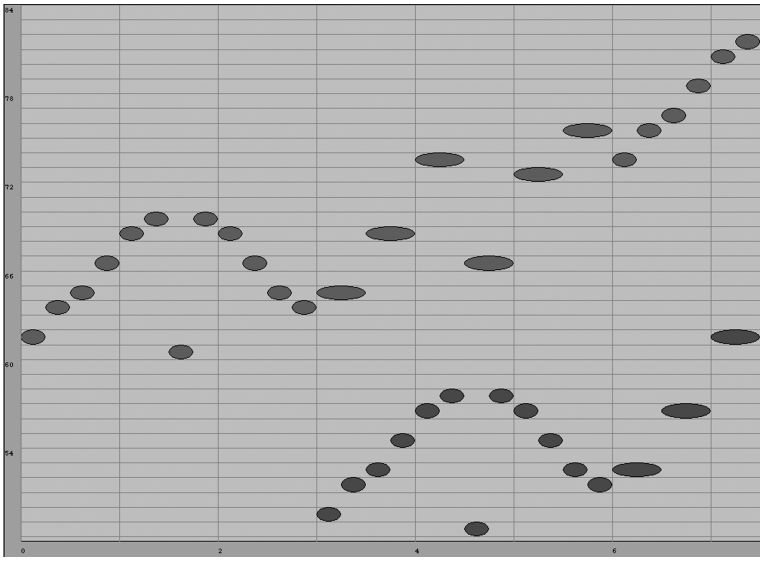

Fig. 4 – Performance layer: MIDI representation for Invention #4

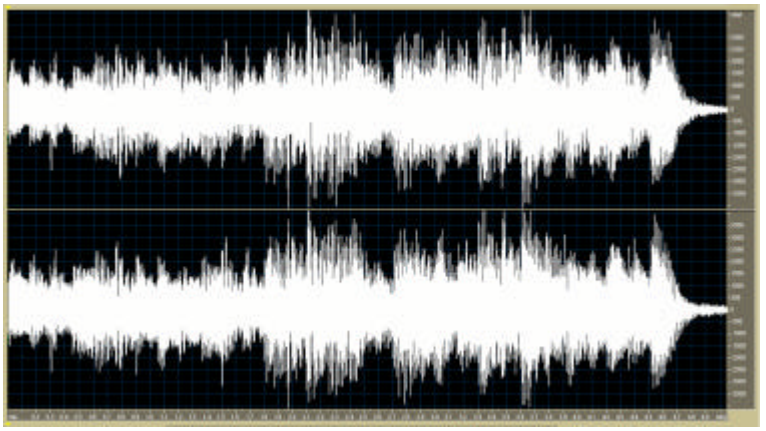

Fig. 5 – Audio layer: waveform representation for Invention #4

MX encoding of the first 5 bars of the piece, without segmentation:

```
<mx>
   <general>
      <description>
         <maintitle>Inventio #4</maintitle>
         <genre />
         <author>J.S. Bach</author>
      </description>
   </general>
   <logic>
      <spine>
         <event id="e0" timing="0" hpos="0" />
         <event id="v1_0" timing="0" hpos="110" />
         <event id="v1_1" timing="256" hpos="300" />
         <event id="v1_2" timing="256" hpos="300" />
         <event id="v1_3" timing="256" hpos="300" />
         <event id="v1_4" timing="256" hpos="300" />
         <event id="v1_5" timing="256" hpos="300" />
         <event id="v1_6" timing="256" hpos="450" />
         <event id="v1_7" timing="256" hpos="300" />
         <event id="v1_8" timing="256" hpos="300" />
         <event id="v1_9" timing="256" hpos="300" />
         <event id="v1_10" timing="256" hpos="300" />
         <event id="v1_11" timing="256" hpos="300" />
         <event id="v1_12" timing="256" hpos="400" />
         <event id="v2_0" timing="0" hpos="0" />
         <event id="v2_1" timing="256" hpos="300" />
         <event id="v1_13" timing="256" hpos="300" />
         <event id="v2_2" timing="0" hpos="0" />
         <event id="v2_3" timing="256" hpos="300" />
         <event id="v1_14" timing="256" hpos="350" />
         <event id="v2_4" timing="0" hpos="0" />
         <event id="v2_5" timing="256" hpos="300" />
         <event id="v1_15" timing="256" hpos="600" />
         <event id="v2_6" timing="0" hpos="0" />
         <event id="v2_7" timing="256" hpos="300" />
         <event id="v1_16" timing="256" hpos="350" />
         <event id="v2_8" timing="0" hpos="0" />
         <event id="v2_9" timing="256" hpos="300" />
         <event id="v1_17" timing="256" hpos="350" />
         <event id="v2_10" timing="0" hpos="0" />
         <event id="v2_11" timing="256" hpos="300" />
         <event id="v1_18" timing="256" hpos="500" />
         <event id="v2_12" timing="0" hpos="0" />
         <event id="v1_19" timing="256" hpos="200" />
         <event id="v1_20" timing="256" hpos="200" />
         <event id="v2_13" timing="0" hpos="0" />
         <event id="v1_21" timing="256" hpos="200" />
         <event id="v1_22" timing="256" hpos="200" />
         <event id="v2_14" timing="0" hpos="0" />
         <event id="v1_23" timing="256" hpos="200" />
         …
      </spine>
   <los>
      <stafflist>
         <staff id="staff0" ossia="no" linenumber="5">
             <Clef type="G" eventref="e0" staffstep="2" octavenum="0" />
```

```
   </staff>
   <staff id="staff1" ossia="no" linenumber="5">
       <Clef type="F" eventref="e0" staffstep="6" octavenum="0" />
   </staff>
</stafflist>
<part id="part0" dfstaffref="staff0">
   <voicelist>
      <voiceitem id="voice0" />
   </voicelist>
   <measure number="1">
      <voice ref="voice0" ossia="no">
         <chord eventref="v1_0" cue="no" grace="no">
            <notehead staffref="staff0">
               <pitchdef staffstep="-1" />
               <duration den="16" num="1" />
            </notehead>
          </chord>
         <chord eventref="v1_1" cue="no" grace="no">
            <notehead staffref="staff0">
               <pitchdef staffstep="0" />
               <duration den="16" num="1" />
            </notehead>
         </chord>
         <chord eventref="v1_2" cue="no" grace="no">
            <notehead staffref="staff0">
               <pitchdef staffstep="1" />
               <duration den="16" num="1" />
            </notehead>
         </chord>
         …
      </voice>
      …
   </measure>
   …
</part>
<part id="part1" dfstaffref="staff1">
   <voicelist>
      <voiceitem id="voice1" />
   </voicelist>
   <measure number="3">
      <voice ref="voice1" ossia="no">
         <chord eventref="v2_0" cue="no" grace="no">
            <notehead staffref="staff1">
               <pitchdef staffstep="4" />
               <duration den="16" num="1" />
            </notehead>
          </chord>
         <chord eventref="v2_1" cue="no" grace="no">
            <notehead staffref="staff0">
               <pitchdef staffstep="5" />
               <duration den="16" num="1" />
            </notehead>
         </chord>
         <chord eventref="v2_2" cue="no" grace="no">
            <notehead staffref="staff1">
               <pitchdef staffstep="6" />
               <duration den="16" num="1" />
            </notehead>
         </chord>
         …
```

```
         </voice>
         …
      </measure>
      …
   </part>

   …
</los>
<notational>
   <graphic_instance filename="inventio4.tif" format="TIFF"
         spine_start_ref="v1_0" spine_end_ref="v1_23">
      <partref partid=""/>
      <rights/>
   </graphic_instance>
</notational>
<performance>
   <performance_instance filename="inventio4.mid"
         spine_start_ref="v1_0" spine_end_ref="v1_23">
      <MIDI format="0">
         <MIDIpartref partid="Piano" track="1" channel="1"/>
      </MIDI>
      <rights/>
   </performance_instance>
</performance>
<audio>
   <clip filename="inventio4.wav" format="PCM" duration="65.127"
         encoding="WAV" freq="44100" nbit="16"
         nchannel="2" spine_start_ref="v1_0" spine_end_ref="v1_23">
      <partref partid=""/>
      <rights/>
      <index time="0.00" measure="1" beat="1" eventref="v1_0/>
      <index time="0.15" measure="1" beat="1" eventref="v1_1/>
      <index time="0.30" measure="1" beat="2" eventref="v1_2/>
      <index time="0.45" measure="1" beat="2" eventref="v1_3/>
      <index time="0.60" measure="1" beat="3" eventref="v1_4/>
      <index time="0.75" measure="1" beat="3" eventref="v1_5/>
      <index time="0.90" measure="2" beat="1" eventref="v1_6/>
      <index time="1.05" measure="2" beat="1" eventref="v1_7/>
      <index time="1.20" measure="2" beat="2" eventref="v1_8/>
      <index time="1.35" measure="2" beat="2" eventref="v1_9/>
      <index time="1.50" measure="2" beat="3" eventref="v1_10/>
      <index time="1.65" measure="2" beat="3" eventref="v1_11/>
      <index time="1.80" measure="3" beat="1" eventref="v1_12/>
      <index time="1.80" measure="3" beat="1" eventref="v2_0/>
      <index time="1.95" measure="3" beat="1" eventref="v2_1/>
      <index time="2.10" measure="3" beat="2" eventref="v1_13/>
      <index time="2.10" measure="3" beat="2" eventref="v2_2/>
      <index time="2.25" measure="3" beat="2" eventref="v2_3/>
      <index time="2.40" measure="3" beat="3" eventref="v1_14/>
      <index time="2.40" measure="3" beat="3" eventref="v2_4/>
      <index time="2.55" measure="3" beat="3" eventref="v2_5/>
      <index time="2.70" measure="4" beat="1" eventref="v1_15/>
      <index time="2.70" measure="4" beat="1" eventref="v2_6/>
      <index time="2.85" measure="4" beat="1" eventref="v2_7/>
      <index time="3.00" measure="4" beat="2" eventref="v1_16/>
      <index time="3.00" measure="4" beat="2" eventref="v2_8/>
      <index time="3.15" measure="4" beat="2" eventref="v2_9/>
      <index time="3.30" measure="4" beat="3" eventref="v1_17/>
      <index time="3.30" measure="4" beat="3" eventref="v2_10/>
      <index time="3.45" measure="4" beat="3" eventref="v2_11/>
```

```
            <index time="3.60" measure="5" beat="1" eventref="v1_18/>
            <index time="3.60" measure="5" beat="1" eventref="v2_12/>
            <index time="3.75" measure="5" beat="1" eventref="v1_19/>
            <index time="3.90" measure="5" beat="2" eventref="v1_20/>
            <index time="3.90" measure="5" beat="2" eventref="v2_13/>
            <index time="4.05" measure="5" beat="2" eventref="v1_21/>
            <index time="4.20" measure="5" beat="3" eventref="v1_22/>
            <index time="4.20" measure="5" beat="3" eventref="v2_14/>
            <index time="4.35" measure="5" beat="3" eventref="v1_23/>
            …
         </clip>
      </audio>
</mx>
```

The same MX file can be processed in order to find repetitive musical patterns and themes. For instance, we can perform a segmentation based on the melodic operator of transposition. In this case, an automatic segmenter (with proper settings) could recognize the two occurrences shaded in Fig. 6.

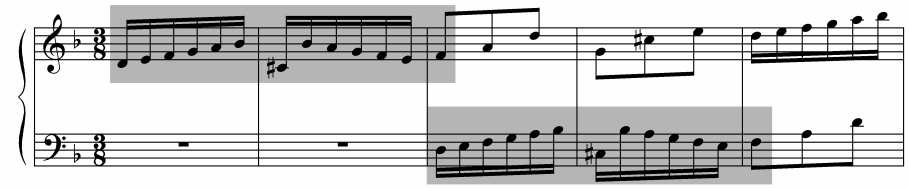

Fig. 6 – The shaded boxes contain the results of a possible segmentation

The resulting MX file would be modified as follows:

```
<mx>
  <structural>
      <themes>
         <theme id="theme0">
            <thm_spineref endref="v1_12" partref="part0" startref="v1_0"
               voiceref="voice0" />
            <thm_spineref endref="v2_12" partref="part1" startref="v2_0"
               voiceref="voice1" />
         </theme>
      </themes>
   </structural>
   <general>
      <description>
         <maintitle>Inventio #4</maintitle>
         <genre />
         <author>J.S. Bach</author>
      </description>
   </general>
   …
</mx>
```

Conclusion

In this paper we discussed two different aspects of music segmentation:

1. the capability to perform an automatic analysis process, based on flexible operators, appropriate data structures, and efficient algorithms;
2. the aptitude to represent the results of (either manual or automatic) segmentation in a comprehensive file format, so that those results can be linked not only to notational level but also to audio, performance, and graphic layers.

Our latest studies in the apparently independent fields of music XML and automatic segmentation originated a very interesting common outcome: the representation of music data and meta-data at any level of abstraction through MX, that revealed to be a comprehensive and effective standard format.

Acknowledgements

The authors wish to acknowledge the partial support of this project by Italian MIUR (FIRB "Web-Minds" project N. RBNE01WEJT_005) and the Italian National Research Council, in the framework of the research program "Methodologies, techniques, and computer tools for the preservation, the structural organization, and the intelligent query of musical audio archives stored on heterogeneous magnetic media", Finalized Project "Cultural Heritage", Subproject 3, Topic 3.2, Subtopic 3.2.2, Target 3.2.1. We also want to acknowledge the members of the IEEE MX WG (PAR1599) for their cooperation and interest in our work.

This work has been made possible by the efforts of researchers and graduate students of LIM.

References

1. *Definition of a Commonly Acceptable Musical Application Using the XML Language*, http://www.lim.dico.unimi.it/ieee/xml.html
2. *LIM Official WebSite*, http://www.lim.dico.unimi.it
3. Bertino, E., Catania B., and Ferrari, E., *Multimedia IR: Models and Languages. Modern Information Retrieval*, Addison-Wesley, 1999
4. Brinkman, A.R., *PASCAL Programming for Music Research*, The University of Chicago Press, Chicago, 1990
5. D'Onofrio, A., *Methods for Integrated Timed Audio and Textual Digital Music Processing*, Master Thesis, Computer Science Department, University of Milan, 1999
6. Frazzini, G., Haus, G., and Pollastri, E., *Cross Automatic Indexing of Score and Audio Sources: Approaches for Music Archive Applications*, Proceedings of the ACM SIGIR '99 Music Information Retrieval Workshop, Berkeley, 1999
7. Guagnini, S., *Strategies and Tools for the Automatic Segmentation of Music*, Master Thesis, Computer Science Department, University of Milan, 1998

8. Haus, G., and Longari, M., *Towards a Symbolic/Time-Based Music language based on XML*, IEEE Proceedings of MAX2002, Milan, 2002
9. Haus, G., and Pollastri, E., *A Multimodal Framework for Music Inputs*, in Proceedings of the ACM Multimedia, ACM Press, Los Angeles, 2000
10. Lerdahl, F., and Jackendoff, Ray, *A Generative Theory of Tonal Music*, The MIT Press, Cambridge, 1983
11. Longari, M., *Formal and software tools for a commonly acceptable musical application using the XML language*, Ph. D. Thesis, Computer Science Department, University of Milan, 2003
12. Ludovico, L.A., *Automatic Identification of Musical Structures: Models and Software Prototypes*, Master Thesis, Informatics Engineering Department, Politecnico of Milan, 2003
13. Polansky, L., *Morphological Metrics*, Journal of New Music Research, vol. 25, pages 289-368, Swets & Zeilinger Publ., Amsterdam, 1996