OntoMusic: from Scores to Expressive Music Performances

Pere Ferrera [a] and Josep Puyol-Gruart [b,1],
[a] *MusicStrands Inc.*
[b] *Artificial Intelligence Research Institute (IIIA).*
Spanish Scientific Research Council (CSIC).

Abstract. The literal performance of the symbols contained in a traditional score is not enough to produce expressive music. Human interpreters use musical knowledge that is not explicitly represented in it. This paper presents a knowledge-based approach to introduce expressiveness to the performance of a score by calculating dynamics and tempo envelopes of the piece, combining implicit musical knowledge with the explicit one contained in the score.

Keywords. Knowledge-based systems, ontologies, applications.

1. Introduction

Apart from the notes and rests, the main musical objects we can find in sheet music are marks or annotations for expressive effects. Mood marks as *vivace*, *maestoso* or *allegretto* instructs for—not very precise— tempo and *emotional* effects. Dynamic marks alter the loudness of the performance: for instance *piano* claims for a quiet sound, or *crescendo* affects the volume of several notes, from soft to loud. Tempo markings as *ritardando* indicate a temporal slow down of the performance. Articulation marks as *legato* say that the notes are to be played smoothly, that is, with no silence in the transition from note to note.

But not *all the music* is explicitly represented in the score [14]. The composer normally annotates only exceptions to the *standard* performance, as the *decrescendo* marks in the first voice of Figure 1. Another example is the introduction of *ritardando* in the final bar of a composition, despite of the lack of the corresponding mark in the score. The clue to this could be the style of the piece: we can introduce it in a folk song but not in a *minuet*.

Our research interests concerns the development of a expressive performance system. We are interested in an ontology-oriented approach [12,10], given emphasis to the structural factors that produce expressiveness [7]. This system uses theoretical musical knowledge and could be complementary to others using learning with performance examples [6].

[1]Correspondence to: Josep Puyol-Gruart, Artificial Intelligence Research Institute (IIIA). Campus UAB. 08193 Bellaterra. Spain. Tel.: +34 935809570; Fax: +34 935809661; E-mail: puyol@iiia.csic.es.

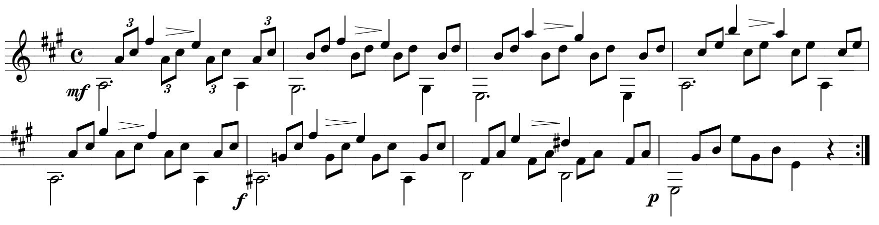


Figure 1. Example of a score with dynamic marks. Excerpt from M. Carcassi (1792–1853) Op.60, Andantino n.3 for guitar.

In Section 2 we present a summary of the ontology used. Section 3 is devoted to the process used to calculate the envelopes for dynamics and tempo. Details on the current implementation and some experiments are explained in Section 4. Finally there are some conclusions and future work in Section 5.

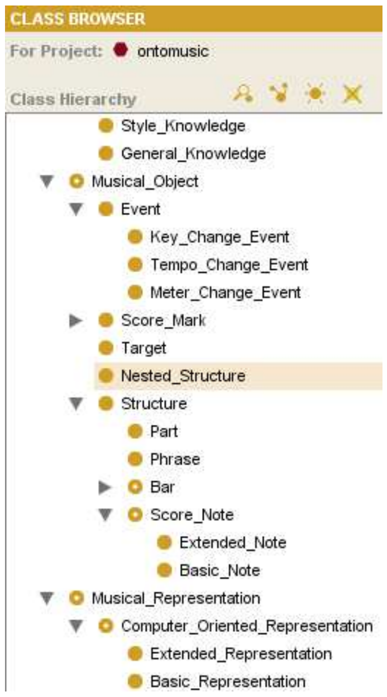
2. An ontology for music

MIDI is the *de facto* standard for the interchange of information among computers and electronic instruments. Its limitations are well known, specially the lack of semantic information. A lot of MIDI extensions has been proposed in [15].

There are other interesting systems for musical representation: from the algebraic approach of Haskore [11] to the Internet-oriented format MusicXML [8]. Some of these systems consider the music as a stream of notes, with the note as the basic object of representation. We are more interested in the high-level structural dimension of pieces, as in the *Object Oriented Music Pieces* [3].

We divide the classes of the ontology into musical objects and musical knowledge (see an example of part of our ontology in Figure 2). We consider musical objects those entities that are contained into a score, like bars and other implicit structural elements: part, phrase and subphrase. We have also defined a hierarchy of nested structures that can be obtained by combination of the others. Segments are also defined to allow the representation of few-notes groups like Narmour groups [6].

We consider musical knowledge those abstract entities that are the basis for any musical analysis, and also the basis for having any structural element, from a simple note (pitch and duration) to high-level knowledge like style knowledge, instrument knowledge, performer knowledge and mood knowledge. The style knowledge is made of constraints that affect structural elements (for instance a blues usually has a 12 phrase length). This knowledge would help us in segmenting phrases, as we will see in Section 3.1. The mood knowledge is a combination of parameters that defines a mood, and these parameters are also related to our structures (for instance, phrasing, meter and harmonic stress are performance characteristics that can define one or other mood). We use this knowledge in our dynamic and tempo envelopes processing as we will see in Section 3.2.



```
        Style_Knowledge
        General_Knowledge
  ▼  Musical_Object
     ▼  Event
           Key_Change_Event
           Tempo_Change_Event
           Meter_Change_Event
     ▶  Score_Mark
        Target
        Nested_Structure
     ▼  Structure
           Part
           Phrase
        ▶  Bar
        ▼  Score_Note
              Extended_Note
              Basic_Note
  ▼  Musical_Representation
     ▼  Computer_Oriented_Representation
           Extended_Representation
           Basic_Representation
```

Figure 2. Example of some classes of the musical ontology.

3. The process

The input of our current implementation is a score with only a melodic line with annotated chords and expressive marks (see the example in Figure 4). This information is processed to complete the ontological objects corresponding to the expressive performance and to calculate the dynamic and tempo envelopes of the piece. In Table 1 you can find the relation among musical analysis, musical factors, and their effects on the envelopes (dynamics and tempo) used in our experimentation.

3.1. Preprocessing: musical analysis and segmentation

We use an automated harmonic and melodic analysis in order to obtain some implicit knowledge like the existence of cadences, the tonal function of chords and the high and low beats.

In order to obtain the implicit phrase and subphrase structures, we are working on phrase and subphrase segmentation using genetic algorithms. The chromosome of our genetic algorithm is a set of bar lengths, each one representing a concrete segmentation (for instance, 8-8-16 is a segmentation of 32 bars). Crossover is a controlled merge of two chromosomes, without almost altering the lengths (because lengths are a crucial characteristic of a good segmentation that should be preserved). Mutation is a permutation of the chromosome (for instance, a possible mutation of 8-8-16 is 16-8-8).

The fitness function for phrase segmentation is based on style knowledge principles, such as common phrase lengths and the importance of the cadence or chord progression.

For instance, blues style uses a rhythmic scheme of eight or twelve 4/4 bars with well-defined patterns for the possible cadences.

Segmenting subphrases is a more difficult task because musicians often find several good segmentations for the same piece and there are no general principles that work like in the phrase case [16]. Therefore, there are some parameters that help, like similarity measures [9]. Our similarity function is based on considering groups of notes with the same duration and melodic tendencies. We obtain good subphrase segmentation when it is based on consider a high similarity degree between all the matched patterns. In fact we are finding repetition patterns. But when trying to extrapolate this to subphrase levels with other granularity, it is better to consider low similarity degrees making of the *difference* the base of the segmentation.

Our similarity function compares two patterns of N bars. We analyze those notes that start at the same beat, and then we use different similarity measures like melodic tendency (whether the melody goes up or down, represented by a binary function), interval tendency (how far is the next note's pitch, or how far was the preceding note's pitch) and rhythmic tendency (what rhythmic value does this note have), in order to obtain a score, that is finally scaled into the interval [-10,10]. Some ideas for this have been taken from [4].

3.2. Processing: envelopes generation

The meter dynamic envelope is deduced from the implicit attributes of notes obtained from the melodic analysis. For the harmonic shapes we are experimenting on deducing the shape of the dynamic and tempo envelope from the tonal function of the theme chords, which are previously deduced from a harmonic analysis of chords. Every chord transition produces a variation on the dynamic, depending on the tonal functions of the chords involved. Chords can also produce tempo variations, as some transitions tend to be introduced by performers with a little *accelerando* or *ritardando*.

To obtain the phrase envelope we use the phrasing principle extracted from general music knowledge: each phrase usually has a climax on the 2/3 of its duration and this climax should imply the higher dynamic level of the phrase. So for each phrase we would have dynamics that follows this principle. The phrase structure also produces tempo variations: if we know that a phrase is conclusive, then we know that it ends on a cadence, and cadences usually lead to a *ritardando*—of course, depending on the style.

The same dynamic principles that in phrase envelope generation can be applied to subphrase envelopes, even though that subphrases don't need to have a climax at all. For simplifying, we make a shape on each subphrase that points a climax the same way we mentioned for phrases. The good results of this process basically deal with the preceding segmentation. Lower subphrase levels are not being exploited right now, mostly because they produce less perceptible results.

Dynamic and temporal envelopes are additively aggregated after assigning weights (see the example in Section 4.1) that depends on the importance of the meter stress, the harmonic stress, the phrasing and subphrasing—envelopes produced by the explicit expressiveness marks in the score are of the highest importance.

analysis/envelope	**Dynamics**	**Tempo**
Phrasing	climax, cadence	climax, cadence
Meter	beats	-
Harmony	tonal function, cadence	chord transition, cadence

Table 1. The current effects considered on dynamics and tempo.

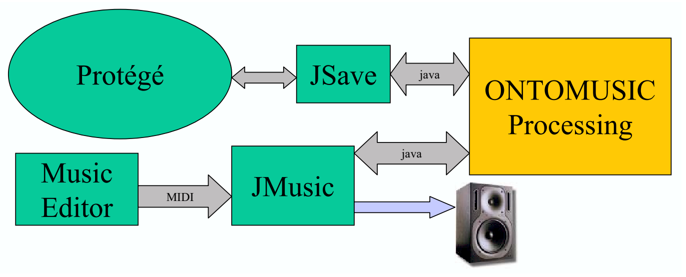


Figure 3. Architecture of the current implementation.

4. Implementation and experimentation

Figure 3 shows the implementation architecture of our application. It mainly consists on a set of free and open source programs based on Java.

- Music editor: it is used to introduce scores and export them in MIDI format.
- Protégé [2]: Protégé is an ontology editor and knowledge base framework based on Java. We use it as a tool to develop the ontology of our domain and as a knowledge base for the instances.
- JSave [17]: It is a plug-in of Protégé. It can be used to generate Java class definition for Protégé classes. The instances of the generated Java classes are also Protégé instances and can use Protégé API calls.
- jMusic [1]: jMusic is a project to provide a library of compositional and audio processing tools. jMusic is implemented in Java and in our project is used for MIDI import/export.
- OntoMusic processing: Our program is written in Java and links to both Protégé (using JSave and the Protégé API) and jMusic API.

Our domain, as explained before, consists in monophonic songs with only a melodic line with annotated chords. The score has to be written with an edition program. To do that, we use a music notation editor with MIDI output. For practical reasons and to control the overall process, we start producing a score that does not contain any expressive mark, because some of the dynamic symbols can affect MIDI playback with unknown effects. Then we have only the notes and rests with their respective durations. After the processing of the MIDI information we manually add the expressive marks contained in the original score and symbolic information of the chords.

The Protégé API allows us to insert, delete or update instances in our knowledge base directly from the OntoMusic processing program. The benefits from this are obvious, for instance, every performance we do is stored in the knowledge base for future reference.

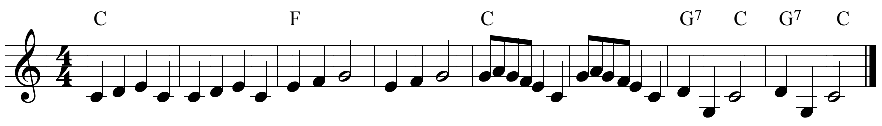


Figure 4. Score of the popular song *Frère Jacques*.

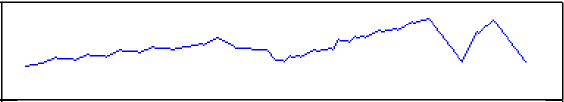
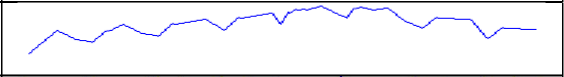
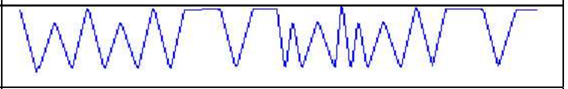
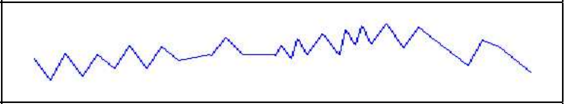
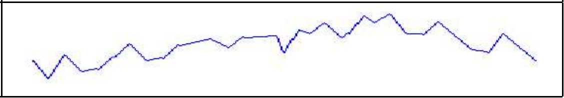
(a)	
(b)	
(c)	
(d)	
(e)	

Table 2. Examples of different envelopes.

To make a clearer link between Protégé and our program, we use Jsave that can generate Java class definitions directly from Protégé classes. These classes are also Protégé API classes, so every change in this object is also a change in the knowledge base objects.

4.1. An example

We show some results on dynamic envelopes through the famous song *Frère Jacques*. This is a simple eight-bar song whose melody and chords are showed in Figure 4. Notice that there are not expressive marks in the score. We can choose different harmonizations for this song, we have chosen this one because the subdominant chord—F is subdominant of C—in bars three and four forces an expressive change.

The phrase segmentation of *Frère Jacques* returns a single phrase, mostly because there is no harmonic cadence before the two last bars. The subphrase segmentation result basically depends on what similarity criteria we use. On one hand, if we require a high similarity between all the segmented subphrases, then the best result is considering each bar as a subphrase. On the other hand, multi-level subphrase segmentation can be obtained just by upper the similarity level we require, as we lower the segmenting level. In other words, shorter subphrases tend to be more similar to each other than longer ones. Anyway, we are focusing on the envelope of one subphrase level, so the first result is useful.

Figures (a), (b) and (c) in Table 2 shows examples of different envelopes obtained with different criteria:

(a) Harmonic stress: The dynamics grow in subdominant (F) and dominant (G) situations. If the chords are assorted, then the harmonic stress dynamic function results in more dramatic performances.
(b) Phrasing (phrasing 50%, subphrasing 50%): We clearly see the subphrase segmentation, and the phrase envelope, with its climax. The phrasing allows us to avoid performing repetitions equally. Performances with high phrasing almost skip the *boring effect* of machine performances.
(c) Meter Stress: Here we can see the high and low beats. Playing with a high meter stress is not usual although it can be used for marches and hymns.

Finally the aggregation of the previous envelopes gives the total dynamics to the song. Only by considering this four parameters (harmony, phrase, subphrase and meter) there are lots of combinations that produce different performances of the same piece. For instance, in Frère Jacques we can make two different combinations that result in different *moods* represented in the envelopes of Figures (d) and (e) in Table 2.

(d) We could obtain a *vivid* performance by setting the meter stress at 40%, the phrasing at 30%, the subphrasing at 30% and the harmonic stress at 10%.
(e) A more calm and *dolce* performance can be obtained by setting the harmonic stress at 40%, the subphrasing at 35%, the phrasing at 20% and the meter stress at 5%.

5. Conclusions and future work

The first experiments with dynamics and tempo envelopes that take into account high-level parameters—like in the Frère Jacques example—show that, with this knowledge approach to musical expressiveness, it is possible to obtain goods results.

With respect to the implementation we plan to change the music edition part by using a more symbolic approach by means of public domain edition software as for instance Lilypont [13]. It could be also interesting to produce a plug-in for Protégé.

It is interesting to extent the domain of the application to more complex musical scores with the inclusion of polyphony [5] and more complex harmony.

We have to revise and complete our ontology, as well as our pre-processing and processing algorithms. The ontology should be richer in high-level knowledge like that about the musical style and particular aspects that depends of the instrument or the performer. We are intended to deeper the study on our segmentation process, as it is the basis for our structural analysis; it seems like the use of fuzzy logic combined with genetics can give good results in the problem of segmenting a piece in different levels of complexity. In a later stage, we plan to combine this knowledge based approach with that of automatic learning in the ProMusic project. Similarity for the segmentation and aggregation—especially for temporal envelopes—will be topics of interest in the future.

Acknowledgments

Authors acknowledge partial support by the Spanish CICYT projects ProMusic (TIC2003-07776-C02-02) and SAMAP (TIC2002-04146-C05-01).

References

[1] *JMusic: Music composition in Java*. http://jmusic.ci.qut.edu.au.

[2] *The Protégé Project*. 2000. http://protege.stanford.edu.

[3] Mira Balaban. The music structures approach in knowledge representation for music processing. *Computer Music Journal*, 20(2):96–111, 1996.

[4] E. Cambouropoulos and C. Tsougras. Influence of musical similarity on melodic segmentation: Representations and algorithms. In *Proceedings of the International Conference on Sound and Music Computing (SMC04)*, 2004.

[5] Darrell Conklin. Representation and discovery of vertical patterns in music. In C. Anagnostopoulou, M. Ferrand, and A. Smaill, editors, *Music and Artifi cial Intelligence: Proc. ICMAI 2002*, volume LNAI 2445, pages 32–42. Springer–Verlag, 2002.

[6] Ramon López de Màntaras and Josep Lluís Arcos. AI and Music: From Composition to Expressive Performance. *AI Magazine*, 23(4):43–57, 2002.

[7] Alf Gabrielsson and Erik Lindström. The influence of musical structure on emotional expression. In Patrik Juslin and John Sloboda, editors, *Music and Emotion: Theory and Research*, chapter 10, pages 223–248. Oxford University Press, 2001.

[8] M. Good. MusicXML for notation and analysis. In Walter B. Hewlett and Eleanor Selfridge-Field, editors, *The Virtual Score: Representation, Retrieval, Restoration*, volume 12 of *Computing in Musicology*, pages 113–124. MIT Press, Cambridge, MA, 2001.

[9] Maarten Grachten, Josep Lluís Arcos, and Ramon López de Màntaras. Melodic similarity: Looking for a good abstraction level. In C. Lomelí and R. Loureiro, editors, *Proceedings of ISMIR 2004: 5th International Conference on Music Information Retrieval*, pages 210–215. Publicacions UPF, 2004.

[10] Nicola Guarino and Christofer A. Welty. An overview of OntoClean. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, chapter 8. Springer Verlag, 2004.

[11] P. Hudak, T. Makucevich, S. Gadde, and B. Whong. Haskore music notation—an algebra of music. *Journal of Functional Programming*, 6(3), 1996.

[12] Mark A. Musen. Ontology-oriented design and programming. In J. Cuena, Y. Demazeau, A. García Serrano, and J. Treur, editors, *Knowledge Engineering and Agent Technology*, volume 52 of *Frontiers in Artifi cial Intelligence and Applications*, pages 3–16. IOS Press, 2004.

[13] Han-Wen Nienhuys and Jan Nieuwenhuizen. Lilypond, a system for automated music engraving. In *Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*, 2003.

[14] John Pierce. Storage and reproduction in music. In Perry R. Cook, editor, *Music, Cognition and Computerized Sound*, chapter 22, pages 285–297. The MIT Press, 1999.

[15] Eleanor Selfridge-Field, editor. *Beyond MIDI: The Handbook of Musical Codes*. The MIT Press, 1997.

[16] Christian Spevak, Belinda Thom, and Karin Höthker. Evaluating melodic segmentation. In *Proceedings of the II International Conference on Music and Artifi cial Intelligence (ICMAI)*, pages 168–182. Springer-Verlag, 2002.

[17] Samson Tu. *JSave*. http://protege.stanford.edu/plugins/jsave.