

# From Music Symbolic Information to Sound Synthesis: An Xml-Based Approach

Goffredo Haus , Luca A. Ludovico and Elisa Russo

Laboratorio di Informatica Musicale (LIM),  
Dipartimento di Informatica e Comunicazione (DICO),  
Università degli Studi di Milano,  
{haus, ludovico, russo}@dico.unimi.it

**Abstract** — This paper deals with the automatic generation of computer-driven performances and related audio renderings of music pieces encoded in symbolic format. A particular XML-based format, namely MX, is used to represent the original music information. The first step is illustrating how symbolic information can originate music performances. The format we have chosen to represent performance information is Csound. Then, an audio rendering is automatically produced. Finally, we will show how the aforementioned computer-generated information can be linked to the original symbolic description, in order to provide an advanced framework for heterogeneous music contents in a single XML-based format.

## I. INTRODUCTION

As regards the process from music symbolic information to sound synthesis, two different aspects are involved:

1. the design and implementation of algorithms to obtain music performances from symbolic information;
2. the representation of both the original symbolic information and the sound synthesis results in a unique suitable format.

The latter aspect is as relevant as the former one, since the description language represents the base algorithms, and its characteristics can deeply influence the process.

As regards music description, let us remark that music communication is very rich; as a consequence, the number of its possible descriptions is high, both from a qualitative and a quantitative point of view. First, the locution “music description” itself can embrace a number of different meanings: the description of a music work could simply mean listing the meta-data about its title, its author(s), its performers, its instrumental ensemble, and so on; another kind of description is based on the symbols that compose the score; finally, also audio/video recordings can be considered music descriptions by full right.

When we refer to the aforementioned qualitative standpoint, we are stating that music-related contents constitute heterogeneous information. Let us cite the examples of opera houses, music publishers, or historical archives (see [1], [2]), where a great amount of heterogeneous music documents are available, e.g. scores, audio/video recordings, iconographic materials and so on.

A comprehensive analysis of music richness and complexity is provided in [3], where six different levels of music description have been identified: *General*, *Logical*, *Structural*, *Notational*, *Performance*, and *Audio* layers. In addition, not only different kinds of media are involved in music description, but each heterogeneous content can be

represented according to different formats. For instance, digital audio and graphical information can be saved in compressed or uncompressed format, with or without loss of information.

This paper deals with the automatic extraction and generation of performance and audio information from a symbolic format. Such information is intrinsically heterogeneous from a qualitative standpoint, and – for each step of the conversion process – it can be encoded in different formats.

For instance, in the application we will present, Csound and PCM are employed for performance and audio information respectively.

The whole computer-based process to obtain automatic audio generation can be modeled by three steps:

1. Description of the original music contents in a suitable symbolic format;
2. Translation of the symbolic score into a suitable performance language;
3. Audio rendering coming from the computer-based performance.

It is worth to remark that our approach allows not only to perform automatically the second and third step, but also to synchronize the heterogeneous contents obtained and to encode them within a single file. As we will explain in the following, the particular format we use allows to start from a score encoded in that format and to add performance and audio information to the file itself.

The result is potentially very rich in multimedia contents, as the three-steps process can be performed a number of times by using different parameters.

The process that brings from score to Csound performances and finally to audio renderings can be useful for many reasons: first, it is possible to generate computer-based performances of a score encoded in any format (in the following, we will introduce an XML-based language), in order to verify possible errors and validate the score itself; besides, it is possible to perform experimentations about timbres and comparisons among different aesthetic results, by using both real-world synthesized instruments and completely invented ones; finally, the automatic generation of performance information (possibly enriched by interpretative models or real-time interaction) can have also didactic and multimedia entertainment implications.

## II. AN OVERVIEW ON THE MX FORMAT

At LIM,<sup>1</sup> a standard language for symbolic music description is under development. This language, known as MX, is a meta-representation for describing and processing music information within a multi-layered environment, in order to achieve integration among structural, score, interpretative, and digital sound levels of representation. Furthermore, the proposed standard should integrate music representation with already defined and accepted common standards.

The development of the MX format follows the guidelines of IEEE P1599<sup>2</sup>. This recommended practice deals with applications and representations of Symbolic Music Information using the XML language. After standardization process, MX will be supported by any kind of software dealing with music information, e.g. score editing, optical music recognition, music performance, musical databases, composition, and musicological applications.

In brief, MX's distinguishing multi-layer structure is composed as follows. The *General* layer is mainly aimed at expressing catalogue information about the piece. The *Logic* layer contains information referenced by all other layers. It is composed of two sub-parts: i) the *Spine* description, used to mark music events in order to reference them from the other layers and ii) the *LOS* (*Logically Organized Symbols*) entity, that describes the score from a symbolic point of view (e.g., chords, rests, etc.). The *Structural* layer contains explicit descriptions of music objects together with their causal relationships, from both the compositional and the musicological point of view. It represents how music objects can be described as a transformation of previously described music objects. The *Notational* layer links all possible visual instances of a music piece. Here MX references the graphical instances containing images of the score. The *Performance* layer links parameters of notes to be played and parameters of sounds to be created by a computer performance. Finally, the *Audio* layer describes audio information coming from recorded performances.

MX has been treated in detail in many papers (e.g., see [3]), and in the following we will describe only the characteristics aimed at automatic sound generation. For further information, working demos and examples please refer to <http://www.mx.dico.unimi.it>.

MX strives for an overall description of music, where the focal point is the single music piece as conceived by its author. MX is intended to describe single pieces, which we define as the most elementary but complete parts a complex composition can be divided into. To provide some examples, a song is an elementary and complete music piece, as well as an aria from an opera or a movement of a symphony.

The representation of music is based on those score symbols typical of Western musical culture, however also non-conventional notations are supported, such as neumes, tablatures, and graphical scores; furthermore, the

encoded piece is not required to have a symbolic description at all. The latter aspect confers further flexibility to MX format.

The concept of comprehensive description of music probably represents the main purpose of MX format. Specific encoding formats to represent peculiar music features, such as audio or symbolic scores, are already commonly accepted and in use; but those formats are characterized by an intrinsic limitation: they can describe very accurately music data or metadata for score, audio tracks, computer performances of music pieces, but they are not conceived to encode all these aspects together. On the contrary, we are interested in a comprehensive description of music, as our purpose is providing a symbolic, a performance-oriented and an audio representation of the same piece.

Besides, we want this comprehensive format to support complete synchronization among time-based descriptions and space-time links towards graphical objects, meaning that audio and video contents are kept synchronized with score advancing. This should happen also when the user switches from a performance to another or from a score edition to another.

In the next section, the main characteristics of MX will be discussed, and the reason why our paper is based on such format should become clear.

## III. MX AS A FORMAT SUITABLE FOR SOUND SYNTHESIS

Let us recall the subject of this paper, i.e. the process that brings from music symbolic information to computer-based performances and finally to sound synthesis. From this perspective, a comprehensive format with the aforementioned peculiarities implies a number of positive consequences:

- When a music piece is encoded in a suitable symbolic format, algorithms can be designed to generate computer-driven performances and audio renderings automatically. The concepts MX is based on allow the extraction of every music event we have to translate or to evaluate. If such events correspond to music symbols according to CWN, they are described in XML format within the *Logic* layer.
- In general, such a format lets symbolic, performance, and audio information be encapsulated within a unique document. In our particular case, it is possible to link the original symbolic information to the derived performance and audio contents.
- Symbolic, performance, and audio information can be described in different encoding formats (e.g., WAV and MP3, Csound and SASL/SAOL, ... ) and in a number of different versions, if needed;
- Symbolic, performance, and audio information can be organized according to a single well-structured multi-layer environment, and they are linked each other. Thanks to this structure, it is possible to implement both a synchronization among different information layers (e.g. between a computer-based performance format such as Csound and the audio produced in MP3 format) and a synchronization among different objects within the same layer (e.g. between an MP3 coming from a computer-driven performance and a WAV file corresponding to a human performance).

<sup>1</sup> Laboratorio di Informatica Musicale (LIM), Dipartimento di Informatica e Comunicazione (DICO), Università degli Studi di Milano.

<sup>2</sup> Institute of Electrical and Electronics Engineers (IEEE) Project Authorization Request 1599 (P1599). In IEEE context, a Project Authorization Request is an official document needed to initiate or change a standard.

The features we have cited provide noteworthy advantages as regards the manipulation of music contents. For our purposes, the most important characteristics of this approach are the following:

1. The possibility to represent and then extract symbolic contents from a well-defined section of the MX encoding.
2. The possibility to add performance and audio information to the original symbolic contents encoded in MX format in order to provide a richer description of the original piece. Multi-layer structure and spine allow to link the original music events to one or more corresponding representations in performance and audio domain, keeping them mutually synchronized. In this way we can have a solid representation of music from every point of view.

#### IV. SCORE ENCODING IN MX

Until now, we have spent many energies to show all the details and capabilities of MX. This preliminary effort is required in order to understand the approach this format can employ to generate sound having only symbolic information. The process of sound generation is strictly connected to the symbolic formalism at our disposal; moreover, if we consider performance as an intermediate step between symbolic representation and sound generation, also the audio description capabilities of the format are interesting from our perspective. MX, thanks to the aforementioned characteristics, allows an advantageous approach to score description, to translation into a performance language and, finally, to audio generation.

From our perspective, score should be intended as an ordered set of music-related symbols. As mentioned before, the original score of the piece is encoded in XML format within the *Logic* layer.

If present, traditional notation is described through a number of music events listed in spine and accurately defined in the *LOS* sub-layer. All the properties of notes and rests are described, from the pitch-related, rhythmic and graphical perspective. In particular, as regards pitch, MX describes both the graphical appearance of a note and the sound we expect.

MX is aimed at a comprehensive representation of music, consequently not only CWN is supported. The representational possibilities offered by MX range from western “classical” music to non-traditional scores (like Indian music or African drumming), from opera to jazz, from pop to rock. It would be virtually possible to reconstruct audio renderings from symbolic information coded also in different notations. In fact, MX supports also non-conventional score representations, such as the graphical ones described in the *Notational* layer. However, at the moment our efforts have been concentrated on the translation of symbols coming from CWN. The automatic translation into a performance language of non-traditional kinds of representation – according to commonly accepted, computer-derived or custom rules – will be the subject of our future research activity.

In MX format, scores that employ CWN or other kinds of symbolic notation are encoded in the *LOS* sub-layer of the *Logic* layer.

Of course, this section cannot provide a complete reference manual, and only the main characteristics of music encoding will be discussed here. However, we think that a short overview of notation in MX can help understanding our approach.

CWN literature demonstrates the necessity to employ at least two different hierarchical structures in music description: i) a structure for staff systems and staves, and ii) another hierarchy for parts, voices, measures, and finally music contents such as chords and rests. MX DTD follows this approach, in order to support the representation of different parts/voices sharing the same staff (e.g. three horns, violin I and II, piccolo and flute), as well as the representation of single parts split on different staves (e.g. piano, harp or organ).

An abstract example of *LOS* contents is shown in Figure 1.

```

<los>
  <staff_list>
    <staff id="sopr_staff">... </staff>
    <staff id="pf_up_staff"> ... </staff>
    <staff id="pf_low_staff"> ... </staff>
  </staff_list>
  <part id="soprano">
    <voice_list>
      <voice_item id="sopr_voice"
        staff_ref="sopr_staff"/>
    </voice_list>
    <measure number="1">
      <voice ref="sopr_voice">
        ...
      </voice>
    </measure>
    <measure number="2">
      ...
    </measure>
    ...
  </part>
  <part id="piano">
    <voice_list>
      <voice_item ref="pf_up_voice"
        staff_ref="pf_up_staff"/>
      <voice_item ref="pf_low_voice"
        staff_ref="pf_low_staff"/>
    </voice_list>
    <measure number="1">
      ...
    </measure>
    ...
  </part>
</los>

```

Figure 1 – The structure of the *LOS* sub-layer.

Before describing music contents in detail, a staff list is provided (*staff\_list* element), where staves are univocally identified. These identifiers will be used later, when music contents are described, in order to associate symbols to the right staff. The other hierarchical level is originated by a number of part elements, each containing a number of voices, described measure by measure.

As regards music notation, for the sake of clarity we will consider the most elementary representation of score symbols in MX: a single note. The graphical example and the corresponding way to encode it in MX are shown in Figure 2. In MX every note is coded through notehead elements. A notehead defines the name and the octave respectively through the attributes *step* and *octave* of pitch

element. The duration is expressed through num and den attributes of duration element, aimed at a fraction representation of rhythmical values.

MX syntax disposes every note within a chord element, where a single note is a degenerate case of a chord (such as in Figure 2). Rests, encoded through rest element, are very similar to chords, but obviously they cannot present pitch and accidental information

Finally, we want to remark that music representation in MX is both score-oriented and sound-oriented. A proof is given by MX accidental notation. A performance-oriented format, such as MIDI, is interested only in how a note should sound, i.e. in its frequency. This is the reason why – in MIDI – B#, C, and D $\flat$  pitches are represented by the same integer value: a computer-driven performance language is not aimed at score representation, but at sound synthesis. On the contrary, a notation-oriented language, such as NIFF, is not interested in the actual note inflecting, but only in what should be printed. Trying to sum up, a two-tier description of accidentals can be outlined: a first aspect implies how a note should sound, no matter if the final result is due to key signature, note accidentals, or execution praxis; a second aspect clarifies which accidental symbols should be printed in a score, no matter if they are required, or if they represent courtesy accidentals or music misspellings. MX, in order to provide a comprehensive description of music, implements both the aforementioned approaches.

#### V. FROM SYMBOLIC FORMAT TO PERFORMANCE LANGUAGE

Now we will discuss the second step of the process described in the introduction, namely the intermediate level between symbolic representation (first step) and audio rendering (third step) of a score. The generation of audio from symbolic information basically depends both on the single notated sounds (e.g. chords, notes and rests) and on the employed timbres. Other aspects can be evaluated, too: for instance, interpretative models can influence both the sounds to be performed and the way they are played.

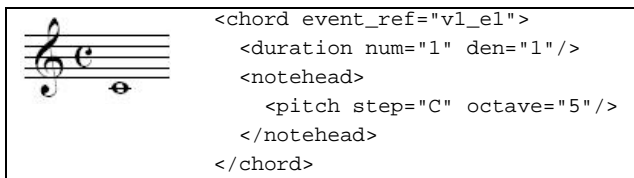
After the identification of a suitable format to represent music contents, we have to choose the tools to be used for sound generation and manipulation. From the encoding perspective, many languages aimed at computer-driven performance can be cited: for example, MIDI [4], and SASL/SAOL [5]. It is worth to remark that they all are supported in MX multi-layer structure. However, the authors decided to employ another performance language, namely Csound, because of its complete and powerful approach to sound generation.

Csound, realized by Barry Vercoe at MIT [6], is a digital synthesizer that permits the simulation of every kind of sound. Music scores encoded in Csound can be played with any timbres, both simulations of real instruments and user-defined ones. The best simulation of existing physical instruments is typically done through physical model synthesis. The other most important models supported by Csound are: additive, subtractive, non-linear distortion, granular, and formant synthesis.

One of the most interesting features of Csound is represented by its intrinsic structure, based on a logical and physical distinction between orchestra and score. The symbolic contents come from the *Logic* layer of MX; the

result of the process will originate Csound code which can be synchronized in MX. Csound takes two formatted text files in input: the orchestra (file .ORC), describing the nature of the instruments, and the score (file .SCO), describing notes and other parameters along a timeline. Then Csound processes the instructions in these files and renders an audio file or a real-time audio stream as output. In this section we will analyze the generation of score files extracting the necessary information from MX's *Logic* layer. The crucial point is how every symbolic event can be translated into sound through Csound instructions without loss of information.

In Csound syntax, every line of a score file represents a single sound event. In particular – for every sound to be produced – the instruments to use, the start time, the duration, the amplitude and the frequency can be specified. Other parameters could be employed, but for our purposes this approach is sufficient. As regards measurement units, time can be expressed in seconds, amplitude in an absolute value and the pitch according to octave-point-pitch-class notation. Once again, other approaches are possible, but this is the best choice for our purposes.



```

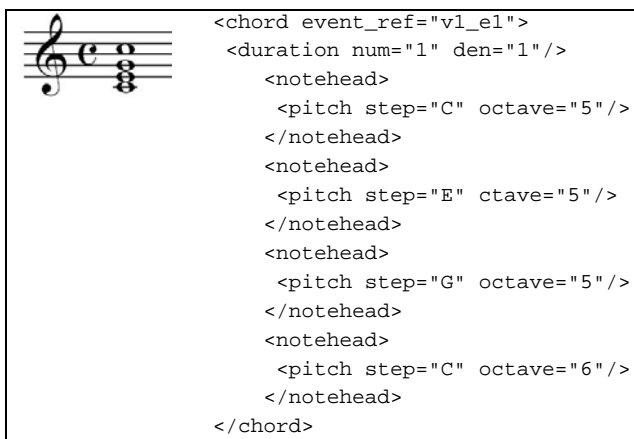
<chord event_ref="v1_e1">
  <duration num="1" den="1"/>
  <notehead>
    <pitch step="C" octave="5"/>
  </notehead>
</chord>
    
```

Figure 2 – A single note encoded in MX.

To make the comprehension easier, first we will study a trivial example, and then we will present more complex ones. As regards the simple case shown in Figure 2, Csound translation is the following:

```
i1 0 4 10000 8.00
```

Now we can illustrate a more complex example. The figure below shows a C major chord together with its MX encoding.



```

<chord event_ref="v1_e1">
  <duration num="1" den="1"/>
  <notehead>
    <pitch step="C" octave="5"/>
  </notehead>
  <notehead>
    <pitch step="E" octave="5"/>
  </notehead>
  <notehead>
    <pitch step="G" octave="5"/>
  </notehead>
  <notehead>
    <pitch step="C" octave="6"/>
  </notehead>
</chord>
    
```

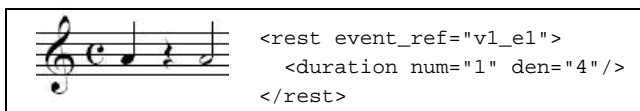
Figure 3 – A chord encoded in MX.

The same events, coded in Csound according to the aforementioned conventions, have the following structure:

```
i1 0 4 10000 8.00
i1 0 4 10000 8.04
i1 0 4 10000 8.07
i1 0 4 10000 9.00
```

In MX every chord is an element which contains a number of sub-elements coding the notes of the chord. On the other hand, in Csound every line represents a single sound event. In this case there are four different sounds whose start time and duration time are the same. More complex cases, i.e. chords composed by notes with different rhythmic values, could be managed by encoding sound events with the same start time for all the notes and different durations.

We will now consider the problem of rest encoding. The figure below illustrates a music example and its XML counterpart.



```
<rest event_ref="v1_e1">
  <duration num="1" den="4"/>
</rest>
```

Figure 4 – A rest encoded in MX.

Usually, in a performance-oriented language, only sounds are coded, so rests are implicitly represented through the start time and the duration time of sound events. In other words, rests are included by the absence of sound.

As a consequence, the Csound code of the above figure results as follows:

```
i1 0 1 10000 8.08
i1 2 2 10000 8.08
```

The different coding of rests in MX and in Csound is a typical example of the different approaches of symbolic and performance languages: in MX rests are explicitly notated by XML elements, whereas in Csound rests result from the absence of sound. In fact, in a symbolic language (like MX) usually we want to code every music symbol of the score. On the other hand, in a performance language (like Csound) we are interested in the information strictly related to the sounds that have to be produced.

The notes of a chord are only some of the contemporary objects used in a score. For example, we can consider simultaneous notes from different parts and voices. In order to translate this in Csound, the algorithm has to reset the counter of the start time whenever a new part/voice begins. The conversion algorithm we have implemented follows the same sequence employed in MX to encode events: parts, measures and voices are managed in this exact order. Our choice implies that the translation in Csound of music symbols belonging to a voice within a measure can be straightforward. Please remember that – according to MX approach – single notes are always considered as degenerate chords, and the translation of simultaneous symbols belonging to a chord (see Figure 3)

is achieved by setting the same start time and the same duration for all the notes.

On the other hand, simultaneous events can occur also because they belong to other voices. When there are different simultaneous parts/voices, the translation is performed measure by measure and the start time of the measure is saved in a variable. As soon as the translation of the single voice ends, the counter is reset to the initial value previously saved.

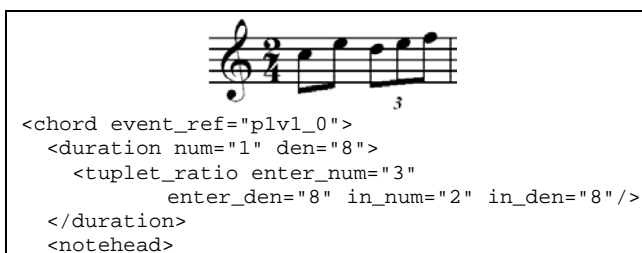


Figure 5 – A polyphonic score where different voices are present.

For instance, in Figure 5 a piano score is shown. While the first measure contains only two voices, corresponding to right hand and left hand respectively, in the third measure three rhythmically independent voices can be recognized. Our algorithm scans the score part by part (but here only piano part is present), measure by measure and finally voice by voice. Consequently, the events in the third measure are encoded in Csound according to the following process: first, the global start time of the measure is saved, then the events belonging to the upper voice are encoded with the proper start time, and when a new voice of measure 3 has to be parsed, the global start time is re-used to synchronize the very first event of that voice.

Basic rhythm information can be retrieved and translated as we have shown in the previous examples, on the base of the information coded in MX. Such information allows a trivial conversion in terms of start time and duration time. However, a detailed management of more complex information, such as different kinds of articulation, tie symbols and irregular groups, implies a more difficult approach. For brevity we are going to treat only the last case, namely tuplets; other examples are shown in the documentation available on-line.

MX specifies the actual duration for every element in an irregular group. Tuplets can be represented considering both the aggregate duration of all the music objects of the group and the single notated duration of every element. For instance, it is possible to represent a situation where 3 quavers take the place of 2 quavers, as shown in the following example. The attributes of `tuplet_ratio` reflect the sentence: “enter 3 quavers in the space of 2 quavers”.



```
<chord event_ref="plv1_0">
  <duration num="1" den="8">
    <tuplet_ratio enter_num="3"
      enter_den="8" in_num="2" in_den="8"/>
  </duration>
  <notehead>
```

```
<pitch step="D" octave="7"/>
</notehead>
</chord>
```

Figure 6 – The encoding of a note in a tuplet.

Thanks to the accurate information provided by MX format, it is possible to convert tuplets and other complex rhythmical layouts into Csound, as shown by the following code example, corresponding to Figure 6.

```
i1 0 0.5 10000 9.00
i1 0.5 0.5 10000 9.04
i1 1 0.33 10000 9.02
i1 1.33 0.33 10000 9.04
i1 1.66 0.34 10000 9.05
```

In conclusion, we have explained how Csound can generate sound events through a symbolic approach based on MX elements. On the other hand, we have also demonstrated that MX can provide sufficient information to a performance language, supplying an intuitive and effective organization of music contents. Soon other applications will be illustrated.

## VI. INTERPRETATIVE MODELS

During sound generation, it can be necessary to consider also those aspects that are not (and cannot be) encoded in the original symbolic format. In other words, the previous section has illustrated some techniques to translate the original symbolic information into a performance language; however, the execution by a human performer is achieved not only by reading the score, but also by interpreting the piece in a unique manner according to the performer's taste and experience.

An approach that takes into account also the human interpretation can be very complex. In this context we only want to point out how many shadings can be produced in an automatic execution to make it credible, and consequently how many parameters should be considered during the design of algorithms suitable to generate sound from symbolic contents. These parameters can be many and various. Now we will only mention the most immediate and basic ones.

At first, we can consider the dynamics notated in the score. These signs typically represent hints for the interpretation and can provide either absolute (e.g. *p*, *mf*, *f*, *sfz*) or relative (e.g. *crescendo*, *diminuendo*) indications. The executor himself can reproduce dynamic symbols in a different manner depending on the role employed in the whole musical piece, historical context, musical genre, personal preferences, etc. Obviously, relative indications cannot be executed in a way independent from the context, since they are related to something else by definition. Please note that even absolute indications usually do not correspond to standard values.

To cite other similar cases, also the metronome of the whole score can be re-interpreted by every performer and usually changes during the piece. We can find similar problems as regards the execution of grace notes, articulations and – in general – every symbol without a fixed rendering in music.

As said before, sound synthesis can be used for different purposes, and not only to generate score renderings similar to human performance. However, in the case we are discussing, we would like that music automatically generated can resemble human performance. As a consequence, the aforementioned problems are relevant and have to be considered in order to create “credible” audio from symbols.

A possible solution consists in allowing *a priori*, real-time or *a posteriori* human intervention to influence the results of score performance. On the contrary, in this context we are interested in computer-based automatic solutions, with the implementation of *ad hoc* algorithms to solve interpretation ambiguity during sound generation. This solution is preferable in order to achieve a completely automatic translation. The application described in the last section implements some basic algorithms to provide a more detailed and credible computer-generated performance. For instance, the absolute dynamic indications are translated by assigning default values within the allowed range, and hairpins (and their equivalents) are realized through appropriate interpolations. The interpretation of time indications and agogics would be a far more complex task. It is sufficient to consider the way tempo markings are encoded: even when standard indications are used, their definition may consist in basic markings (e.g. *Allegro*, *Andante*, *Adagio*), common qualifiers (e.g. *Assai*, *Con moto*, *Con brio*), mood markings (e.g. *Vivace*, *Maestoso*, *Sostenuto*), and even terms for change in basic tempo (e.g. *Ritenuato*, *Stretto*, *Rubato*). All this information is encoded in MX, but in general its semantic content cannot be inferred automatically.

As regards all the aspects whose meaning is not codified or it is difficult to be interpreted by a machine, a credible human-like performance can be simulated by formalizing some typical human behaviors.

## VII. AUDIO RENDERING AND SYNCHRONIZATION

In the previous sections we have analyzed how to encode some performance parameters from the symbolic information of MX format. The last step of the process, namely the transformation from computer-based performance to sound, can be based on Csound itself: in fact, this software tool allows to reproduce and save the results of sound synthesis in digital format.

As said before, there exist well-known and commonly accepted standards to represent both performance and audio information. In particular, for our purposes we have employed Csound as the reference language for performance and PCM format for audio. Let us recall that such formats are supported by MX, in the sense that they are linkable from the proper MX layers. In other words, thanks to MX characteristics, it is possible to enrich the original file (that presented only symbolic information) with the results of the process that has originated a computer-based performance and finally an audio rendering.

As regards synchronization, we have affirmed that different layers can be connected each other through the spine structure; as a consequence, every music event (described from a symbolic perspective in the *Logic* layer) can be linked to its representation in the layers devoted to

computer-driven performance and audio [7]. Synchronization implies:

1. the recognition of each music event listed in spine inside performance and audio files;
2. the update of the original MX file in order to encode also synchronization information.

In general, when we want to create an MX file where not only symbolic information is present, but performance and audio contents as well, a problem arises: it is still difficult to obtain automatic synchronization in performance files and audio tracks, even if the complete score is known.

Such problem can be easily overcome when performance and audio information are automatically generated by a computer system that parses the original score. In fact, in this case it is sufficient to keep trace of the process to have all the information required. For example, in a Csound score the start time and the duration of each sound event are clearly identified. An algorithm that keeps trace of these values and associates them to the corresponding audio events easily achieves a correct synchronization.

According to MX multilayer structure, the three steps constituting the automatic generation process are represented in the *Logic*, *Performance*, and *Audio* layers respectively. All these representations have to be connected to spine; in this way, each music event is linked to its logical, performance and audio rendering.

As regards the *Performance* layer, it is sufficient to relate the identifier of spine events to the corresponding line in Csound score file. An example is provided in Figure 7.

```
<performance>
  <csound_instance>
    <csound_score file_name="C:\adagio.csd">
      <csound_spine_event line_number="3"
                           event_ref="plv1_1" />
      <csound_spine_event line_number="4"
                           event_ref="plv1_2" />
      ...
    </csound_score>
  </csound_instance>
</performance>
```

Figure 7 – Synchronization of a Csound score in MX.

After obtaining a Csound score and associating timbres to parts/voices through a Csound orchestra, finally a waveform can be generated. In Csound score the start time and the duration of each sound event is known, so this information can be used to achieve synchronization between the music symbols in the *Logic* layer and their rendering in the *Audio* layer. For every track event – namely events mapped in audio tracks – MX encodes the current reference to spine and the absolute time expressed in absolute terms (see Figure 8).

```
<audio>
  <track file_name="C:\adagio.wav" ... >
    <track_indexing timing_type="seconds">
      <track_event timing="0"
                   event_ref="plv1_1" />
```

```
<track_event timing="0.1875"
             event_ref="plv1_2" />
...
</track_indexing>
</track>
</audio>
```

Figure 8 – Synchronization of an audio clip in MX.

As we have said before, thanks to MX it is possible to implement both a synchronization among different information layers (e.g. between a computer-based performance format such as Csound and the audio produced in MP3 format) and a synchronization among different objects within the same layer (e.g. between an MP3 coming from a computer-driven performance and a WAV file corresponding to a human performance). We can refer to the former as inter-layer synchronization and to the latter as intra-layer synchronization, where inter-layer synchronization is characterized by linking heterogeneous media contents, whereas intra-layer one creates mappings among different encodings of contents of the same type. As a consequence, we can obtain one or more computer-based performances from the original symbolic information, and one or more audio renderings of such performances, and everything will be kept synchronized.

In conclusion, our approach establishes both a quantitative and a qualitative enrichment of the original music information contained in an MX file. The qualitative enrichment is achieved by adding heterogeneous types of multimedia information, whereas the quantitative enrichment is due to a number of media objects of each type that are virtually linkable to the original file.

### VIII. A CASE STUDY

Thanks to its characteristics, MX format is extremely flexible: it can be applied to classical music as well as Indian raga, to jazz as well as contemporary music, to neumes as well as non-conventional music notations [8].

In order to apply all the concepts shown in this paper and to demonstrate their implementability, at LIM we have designed and developed an application which carries out the translation from symbolic to audio information by using an intermediate performance language and focusing on CWN. This application loads an MX file and shows it in the left part of the interface. Therefore, the user has to choose a Csound file that contains the orchestra and the score. This file is visualized in the right part of the main window. For the sake of easiness, we have chosen the single Csound file format based on XML (namely .cds), so that both the orchestra and the score are managed within one document.

A number of human interventions and user settings is allowed. For instance, we can cite the choice of instruments and their association to parts/voices, the choice of base dynamics, the fixing of the execution time, the decision to write or not data about synchronization in the original file.

Before starting the process, two files are required: the MX file containing a valid score (which corresponds to the first step of the aforementioned process) and a .csd file

with the Csound orchestra to be completely or partially used.

The application performs the second and the third step mentioned in the introduction: when the MX score is loaded, it is translated into a Csound score and added to the .csd file that contains the timbres. The synchronization information between symbolic and performance is added to the original MX, and the MX view is refreshed. Therefore, the .csd file is rendered by CsoundAV.exe and the resulting PCM file can be listened to through a special button. The further synchronization data are added to MX.

This software, besides demonstrating the concepts we have illustrated, has some practical implications. First, it permits every kind of timbre experimentation, with any piece of any music genre. Experimentations were made during the past years with classic pieces or with pieces purposely written with aesthetic goals. In this sense, our application supports an easy, direct and immediate approach that electronic musicians can adopt to study timbre manipulation. Besides, the application can also become a useful didactic instrument to learn the basics of timbre study or to experiment synthetic sound effects. Finally, this instrument supports the validation of scores directly written in XML or converted in MX by filters, as it makes symbolic scores sound with no human intervention.

#### IX. RELATED WORKS AND CONCLUSIONS

MX format provides an innovative and comprehensive way to describe music information. As regards the content types treated in this paper, of course there exist many other formats able to convey score, performance and audio information, but they all provide only a partial view of the big picture and their data are often uncorrelated. On the contrary, the MX format can describe heterogeneous information in a unique document and in a synchronized fashion. This is the main reason why MX-based applications can be effective tools to perform the process that brings form score to performance and audio.

#### REFERENCES

- [1] G. Haus, "Rescuing La Scala's Music Archives", *Computer*, vol. 31, no. 3, pp. 88–89, 1998.
- [2] G. Haus and L.A. Ludovico, "The Digital Opera House: an Architecture for Multimedia Databases", *Journal of Cultural Heritage*, vol. 7, no. 2, pp. 92–97, 2005.
- [3] G. Haus and M. Longari, "A Multi-Layered, Time-Based Music Description Approach Based on XML", *Computer Music Journal*, vol. 29, no. 1, pp. 70–85, 2005.
- [4] E. Selfridge-Field, "Beyond MIDI: the handbook of musical codes", MIT Press, 1997.
- [5] E.D. Scheirer and B.L. Vercoe, "SAOL: The MPEG-4 Structured Audio Orchestra Language", *Computer Music Journal*, vol. 23, no. 2, pp. 31–51, 1999.
- [6] R. Boulanger, "The Csound book", Cambridge, MIT Press, 2000.
- [7] A. Baratè, G. Haus, and L.A. Ludovico, "An XML-Based Format for Advanced Music Fruition", *Sound and Music Computing 2006*, 2006.
- [8] D.L. Baggi, A. Baratè, L.A. Ludovico, and G. Haus, "A computer tool to enjoy and understand music", *EWIMT 2005*, pp. 213–217, 2005.