Computer Printing, Storage, and Transfer of Musical Scores

Gerard Assayag
ACT Informatique, 12 rue de la Montagne St. Genevieve, 75005 Paris.

Dan Timis
Department of Music, University of California, Santa Barbara CA 93106.

ABSTRACT

In our paper A ToolBox for Music Notation presented at the 1986 ICMC in the Hague, we divided the problem of music printing by computers in three levels:

1. Design and manipulation of elementary graphical objects. At this level, displaying routines with regard to absolute position on the graphical surface must be provided for a multiplicity of printers and plotters.

2. Abstract relationships (connections and spatial dependencies) among musical objects are expressed.

3. Layout, expert rules, interactive programs, and other high level applications can be implemented.

Different approaches at level three could communicate with multiple implementations at level one via level two. We presented MUSCRIPT an object oriented extension to the POSTSCRIPT language. Providing tools for relative positioning of musical objects, MUSCRIPT, is one possible implementation of level two.

In the present paper, rather than introducing a particular program in a particular language, we generalize our experience and present several theoretical ideas. We try to define what are the most important attributes of a musical score and the relationship between the score and the musical structure. We try also to define what are the essential elements of a score for manipulation and storage on a computer and for communication between machines.

At last we present a model for coding musical scores.

What is a score?

New ways of producing, articulating, and structuring sound, in contemporary music, lead to the use of new musical signs and new ways of representing music, thus redefining the notion of score. A printing program should not impose any limitation on the composer's imagination, enabling any kind of innovation in the field of notation. Written by hand, engraved, or printed with the computer, the score as means of communication between musicians, remains an important tool.

The notion of score was also enlarged by the use of computers in music synthesis. The input of many programs implementing sound synthesis algorithms is often called a score. Although very important in the computer music area, any method of communication from a musician to a machine does not concern the present paper. A computer application implementing a composition algorithm could generate a graphical representation of the result. In this case the communication takes place from the machine to the user.
A score is that a graphical representation of a piece of music, in the form of a collection of musical signs on a surface like paper or a computer screen. It can be produced by a person or a machine but its destination is always the musician.

Musical Structure, Score Structure, and Associated Semantics

It is difficult to give a good and complete definition to the notion of Musical Structure. We could consider it as a complex network of relationships between musical events. An important observation is that we tend to attribute a unique structure to a musical piece, but several possible score representation (by different publishers for example), and a lot of interpretations (by different performers).

By Score Structure we denote a collection of musical signs and dependencies between them. The meaning of each sign and the relationship between Musical Structure and Score Structure is established through the associated Semantics which could be viewed as a set of translation rules from one type of structure to the other.

A simple way of representing a piece of music could be the use of a list of temporal events. Let us suppose that a musical piece is a collection of events, each with only two attributes: pitch and duration. We will call it the Simplified Musical Structure of a piece. As an example we could have a (B, quarter note) event followed by a (G sharp, half note) one.

The Score Structure could be a collection of staves, clefs, notes, beams, etc. For example a treble clef associated to the second line of a staff, is followed by a quarter note head associated to the third line of the same staff; a stem could be associated to the note, etc.

A possible associated Semantics would contain rules such as:
- A quarter note head associated to the third line of a staff when to the right of the note the closest clef associated to the same staff is a treble clef, represent an event whose pitch component is a B.
- A quarter note head associated to a stem when there is no flag or beam associated to the stem, represent an event whose duration component is a quarter note.

There is always a very large number of possible graphic representation of the same musical structure. Although most of the score structures have a unique interpretation in a given associated semantics, some scores are ambiguous.

There is no ambiguity in translating the following musical fragment into simplified musical structure.

![Musical fragment](image)

Printing the same fragment using only the result of this translation would probably produce the following representation which is very different from the initial one.

![Simplified representation](image)

Of course, we could increase the amount of information contained in each event. How many attributes do we need to add in order to ensure that any score translated into musical structure and back, will remain unchanged? Probably we will end by putting all the information regarding the musical structure plus all the information concerning the score structure. We will have more information than the score structure alone and will lose its ambiguity.

A parallel with the written text could be made. A text processing application would treat the text as a collection of letters grouped together in words separated by spaces and newlines. Nobody would think to translate the words and then to manipulate and store meanings, then again to translate them back into words. Most of the Macintosh musical application manipulate pitches and durations instead of note heads, stems and beams. This explains why their score rendering capabilities are considered too primitive and too idiomatic by many musicians.

Common-practice Music Notation (CMN)

The general opinion is that notation that makes use of staves, clefs, notes, etc, is CMN. The following example borrows its semantics from
Stockhausen's Klavierstück X. An ascending beam means *accelerando*, the descending one denotes *ritardando*. Thick notes designate initial tones or destination tones of melodic groups. Slurs connected by a slur mean sustain until the end of the slur.

A distinction should be made between the two components of CMN: Common-practice Musical Symbols and their associated Common-practice Semantics. Rather than the introduction of new symbols, associating a new semantics to musical signs is more likely to lead to New Music Notation. Adding signs like quarter-tones but keeping most of the associated common-practice semantics leads to something which could be called Extended Common-practice Music Notation. Of course many contemporary music pieces make use of both new symbols and new semantics.

Our first version of MUSCRIPT was limited in the number and the type of symbols by the use of the SONATA font, the only POSTSCRIPT musical font available. As there is no associated semantics to each sign or to each connection, MUSCRIPT is a general tool producing any type of notation, not exclusively CMN like some people could think.

Musical Symbols, Connections and Dependencies

In our previous paper we divided musical symbols (objects) into two categories:

- iconic symbols that can only be scaled or translated (notes, clefs, accidentals, etc.)
- algorithmic objects that beside scaling and translation, support other types of transformations, like stretching (staves, beams, stems, slurs, etc.)

Each symbol can have a number of attributes like size for a clef, length for a stem, etc. Relationships between the different symbols in a score are very complex and their accuracy is essential.

Connections could be defined as follows: a particular attribute of an object coincides with a particular attribute of another object. As examples connecting a stem to a note means having an exact correspondence between a particular point of the note and a particular point of the stem; connecting a note to a stave means that the horizontal axis of the note coincides with a particular line or space of the stave. Connections do not have attributes.

A dependency is another type of relationship between two objects, which could have attributes. For example a note follows another note: the attribute could be the distance between two notes. The attributes of the dependencies when present can be changed without affecting the structure of the score. In the next example the distance between the notes is smaller and the distance between the staves is larger. The attributes of each symbol and the connections did not change.

Formated and Unformatted Score Structure

The analogy between scores and text could be useful. Unformatted text is a long list of symbols (characters) each with his own attributes (like font, size, etc.). There is only one type of spatial dependency: a character follows another character (no attribute like the distance between two consecutive symbols, is present).
Formatting the text means introducing new elements (like pages, left and right margin, paragraph indent, etc.), giving attributes to the spatial dependencies (the horizontal distance between two consecutive symbols), relating symbols to the new elements, etc.

We could consider an unformatted score as a collection of symbols, connections, and dependencies without attributes. The relationships are both horizontal and vertical, but there is no information about the exact distance between symbols. It is an abstract representation that an exact graphical example is impossible. If we set the attributes of the spatial dependencies of the first example, to a very small value we obtain:

```
\begin{music}
\input{example.music}
\end{music}
```

Formatting a score is similar to formatting text. First we introduce pages, left and right margins, etc. Then we give values to the attributes of spatial dependencies, we cut when we reach the right margin or when we find convenient and start again from the left margin, lower or on another page.

Score Storage and Transfer

We believe that the best way to store scores on a computer and to transfer them between machines is in the form of Unformatted Score Structure. We have now all the elements to define a coding model. One should start by enumerating the musical symbols and their attributes, when different from the default; then connections and spatial dependencies.

The last example (the unformatted version of the first) has 15 symbols:

- 4 notes
- 4 stems
- 2 flags
- 1 beam
- 1 bar

The spatial dependencies are exclusively horizontal.

From left to right:
- the clef follows the left margin of the stave
- the time signature follows the clef
- the first note follows the time signature
- the second note follows the first one
- the third note follows the second one
- the fourth note follows the third one
- the bar follows the fourth note

The connections are:
- the clef with the stave (on the vertical axis)
- the time signature with the stave (on the vertical axis)
- each note with the stave (on the vertical axis)
- the upper and lower ends of the bar with the stave (on the vertical axis)
- the left end of the stave with the bar (on the horizontal axis)
- each stem to a note (both horizontal and vertical)
- each flag to the first and fourth note (both horizontal and vertical)
- the left end of the beam to the second stem (both horizontal and vertical)
- the right end of the beam to the third stem (both horizontal and vertical)

References:
