Using Advanced GUIDO as a Notation Interchange Format

Holger H. Hoos[*]
Department of Computer Science
University of British Columbia
2366 Main Mall
Vancouver, BC, V6T 1Z4, Canada
`hoos@cs.ubc.ca`

Keith A. Hamel
School of Music
University of British Columbia
6361 Memorial Road
Vancouver, BC, V6T 1Z2, Canada
`hamel@interchange.ubc.ca`

Kai Renz
Department of Computer Science
Darmstadt University of Technology
Wilhelminenstr. 7
D-64283 Darmstadt, Germany
renz@iti.informatik.tu-darmstadt.de

Abstract

GUIDO Music Notation is a new music representation format based on the notion of representational adequacy, i.e., it represents simple musical concepts in simple ways, and requires complex representations only for complex scores. This paper shows how Advanced GUIDO Music Notation, the second layer of the GUIDO design, can be used as an adequate and complete notation interchange format. We demonstrate how all page layout, positioning, and spacing information can be easily specified in GUIDO so that all details of a score are exactly represented. Several implementations of Advanced GUIDO are discussed, including the music notation software NoteAbility and the GUIDO NoteServer, an on-line notation engine on the WWW. These first implementations of full GUIDO support demonstrate that Advanced GUIDO can be used to efficiently and accurately represent and exchange musical score information ranging from simple melodies to complex orchestral pieces.

1 Introduction and Background

There have been many attempts to specify and establish an interchange format for music notation. In comparison with other notation interchange formats such as NIFF, SMDL, *cmn*, and DARMS (Grande, Belkin; 1996; Sloan; 1997; Selfridge 1997), the recently introduced GUIDO Music Notation (GMN) format offers a unique combination of the following features (Hoos *et.al.*; 1998a): it is

- representationally adequate;
- human readable;
- platform independent;
- easily extensible;
- easy to implement.

Representational adequacy refers to the ability to represent simple musical concepts in simple ways, and to require complex representations only for complex scores. GMN uses a text-based representation, like HTML or XML, so it is both humanly readable and platform independent. GMN has a three layered design. The first layer (Basic GUIDO) includes standard musical notions (Hoos, Hamel; 1997). The second (Advanced GUIDO) supports exact score formatting and more sophisticated musical concepts. The third layer (Extended GUIDO) covers features which are beyond conventional music notation (e.g., microtonal information, absolute timing, or hierarchical scores). GUIDO is built on a clean and simple syntactical framework which makes it easy to implement GUIDO support in new or existing applications and also facilitates customizations that might arise in specific applications or research projects.

[*]Corresponding author; also affiliated with the Department of Computer Science, Darmstadt University of Technology, Wilhelminenstr. 7, D-64283 Darmstadt, Germany.

The remainder of this paper is structured in the following way: In Section 2 we introduce Advanced GUIDO Notation and highlight its various features, especially with respect to exact score formatting and layout. Then, in Sections 3 we describe two notation applications supporting GUIDO Music Notation: GUIDO NoteViewer/Server, a stand-alone program for graphically displaying and printing scores specified in GUIDO which is also available via a platform independent online service on the WWW; and NoteAbility, Keith Hamel's interactive notation editor, which is currently available (in different versions) for several platforms, including Mac OS-X and Windows 95/98/NT. Finally, in Section 4 we summarise the main aspects of using Advanced GUIDO Music Notation as a notation interchange format, highlight some of its main differences when compared to existing approaches, and briefly outline ongoing and planned research and development projects.

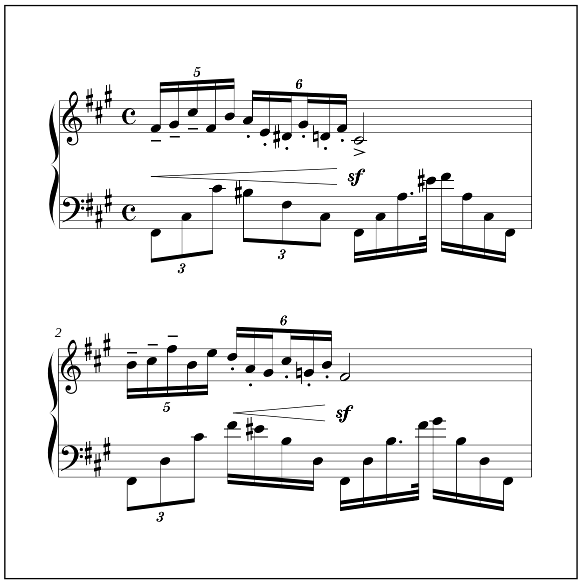
2 Advanced GUIDO Music Notation

Advanced GUIDO Notation comprises some of the more advanced concepts not covered in Basic GUIDO Notation, such as glissandos, arpeggios, clusters, different types of noteheads, different types of staves, and many features from contemporary notation. It also addresses issues of advanced score formatting such as exact spacing and positioning of notational and graphical elements. Using Advanced GUIDO, it is possible to specify exact score-formatting information that can theoretically be used by any kind of professional notation software. This feature makes Advanced GUIDO an ideal candidate for a platform- and application-independent notation interchange format.

2.1 Syntax extensions

Advanced GUIDO is based on the same syntax as Basic GUIDO with three slight extensions: named tag parameters, parameter values with units, and multiply overlapping tag ranges.

Named tag parameters. In Advanced GUIDO, tag parameters can be specified by using predefined names. An example for this is `\clef<type="g2", size=0.5>`, which specifies a g-clef on the second



```
{[\pageFormat<w=151mm,h=151mm,lm=14mm,tm=25mm,rm=12mm,bm=27mm>
  \systemFormat<staves="1-2",dx=-3mm>
  \accol<id=0,range="1-2",style="curlyBrace">
  \staff<id=1,dy=16.8mm> \staffFormat<size=1mm>
  \stemsUp  \space<4mm> \clef<"g2"> \space<6.5mm> \key<"f#">
  \space<8mm> \meter<"C"> \space<7mm>
  \crescBegin<dx1=-1mm,dy1=11.5mm,dx2=-1mm,dy2=11.5mm,h=4mm>
  \beam<dy1=11hs,dy2=9hs>(\ten(f#1/20) \space<5mm> \ten(g#)
    \space<5mm> \ten(c#2) \space<5mm> f#1 \space<5mm> h)
  \space<5mm>
  \beam<dy1=7hs,dy2=8hs>(
    \beam( \stacc(a1*1/24) \space<4.5mm> \stacc(e)
      \space<6mm> \stacc(d#) ) \space<4.5mm>
    \beam( \stacc(g#) \space<6mm> \stacc(d)
      \space<4.5mm> \stacc(f#) ) ) \crescEnd \space<4.5mm>
  \intens<m="sf",dx=-1mm,dy=12mm> \accent( c#1/2 )
  \space<4.55cm> \bar<2>
  \newSystem<dy=6.55cm> ...],
[ \staff<id=2> \staffFormat<size=1mm>
  \stemsDown \space<4mm> \clef<"f4"> \space<6.5mm> \key<"f#">
  \space<8mm> \meter<"C"> \space<7mm>
  \beam<dy1=-7hs,dy2=-15.8hs>( f#-1/12 \space<8.5mm> c#0
    \space<8.5mm> c#1 ) \space<8.5mm>
  \beam<dy1=-11.8hs,dy2=-7hs>(h#0 \space<10mm> f#
    \space<10mm> c#)
  \space<9mm> \beam<dy1=-7hs, dy2=-17.4hs>(
   f#-1*1/16 \space<6mm> c#0 \space<6mm> a.
    \space<8mm> \beam(e#1/32)) \space<4mm>
   \beam<dy1=-18.2hs,dy2=-7hs>( f#1/16 \space<6mm> a0
      \space<6mm> c# \space<6mm> f#-1 )
   \space<5mm> \bar<2> \newSystem  ...] }
```

Figure 1: Short excerpt from Scriabin's Etude Op. 8 No. 2 (50% reduced in size) and corresponding, slightly abbreviated, Advanced GUIDO description.

line (i.e., a treble clef) which has only 50% of its standard size. Note that parameter names are optional and can be omitted, as long as all parameters are specified. Using parameter names not only increases the readability of Advanced GUIDO sources (which is beneficial for implementing GUIDO support), it also makes it possible to only partially specify parameters, and to assume default values for unspecified optional parameters.

Parameter values with units. Advanced GUIDO allows different measurement units to be used with tag parameters. These units, such as `cm`, `mm`, `in` (inches), `pt` (points), and `pc` (picas) can be optionally used with the numerical values of sizing and positioning tag parameters by specifying the unit type directly after the value. As an example, consider the tag `\pageFormat<20cm,40cm>`, which defines the page format to be used to specify the score as 20cm wide by 40cm high.

Multiply overlapping tag ranges. Basic GUIDO already supports overlapping tag ranges by using begin / end tags, such as in `\slur(c1/2 \crescBegin e/8 d) g/4 \crescEnd`. This mechanism, however, does not handle the (rather rare) cases of multiple occurrences of the same tag with overlapping ranges. To support the most general case of arbitrary multiply overlapping tag ranges, Advanced GUIDO supports explicit disambiguation for begin / end tags. This is done by allowing the begin / end tags to be extended with a unique numeric postfix, using the syntax `\*Begin:`n`<...> ... \*End:`n. Thus, two overlapping slurs can be represented as `\slurBegin:1 ... \slurBegin:2 ... \slurEnd:1 ... \slurEnd:2`.[1]

[1]Note that this example can alternately be realised without using disambiguated begin / end tags; however, more complex examples, like three mutually overlapping slurs, require explicit disambiguation.

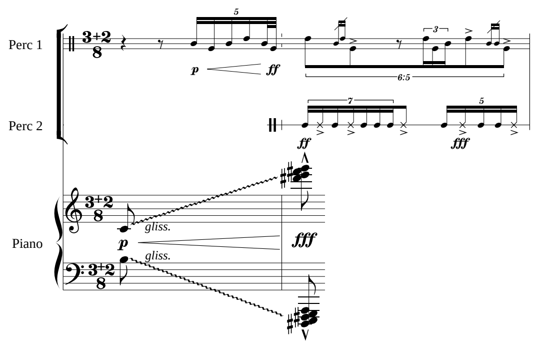
2.2 Exact formatting and score layout

Advanced GUIDO allows you to completely specify the formatting and layout of scores. This is realised by including some new tags and a large number of additional tag parameters to existing Basic GUIDO tags which control the physical dimensions of score pages, the position and size of systems and staves, as well as the exact location of all graphical elements associated with staves. While notes, rests, most notational symbols, and text are typically associated with staves, Advanced GUIDO also supports graphical and text elements which can be placed at specific locations on the page, independent from staves. While the full set of mechanisms and corresponding tags for completely specifying score layout and formatting cannot be presented here in detail, we will highlight some of the main components and briefly outline others.

A small set of tags is used to specify the physical dimension and margins of score pages (`\pageFormat`), the relative positioning of systems on a page (`\systemFormat`), type and size of staves in a sytem (`\staffFormat`) and their relative location (`\staff`), page and system breaks (`\newPage`,`\newSystem`), etc. Spacing and layout are based on graphical reference points, which Advanced GUIDO specifies for each notational element. Some elements (such as systems, or composer and title information) are positioned relative to the page, others are relative to the staves or notes they are logically associated with.

Generally, for elements associated with a staff, vertical offsets are usually specified relative to the size of the staff, using the relative unit halfspace (`hs`).[2] Likewise, the size of many elements, such as notes or clefs, is always specified relative to the size of the staff. The most important mechanism for exactly specifying horizontal locations is the `\space` tag. Placed between

[2]One halfspace is defined as the difference in vertical position between two adjacent notes, e.g. between `c` and `d`.



```
{[ (* voice 1*) ... \systemFormat<staves="1-4",dx=0>
  \accol<id=0,range="1-2",style="straightBrace"> ...
  \staff<id=1,dy=2.5cm> \barFormat<style="staff">
  \staffFormat<style="3-line",size=0.875mm> ...
  \clef<"perc"> ... \meter<sig="3+2/8"> ... \dottedBar<2>
  ... \tuplet<format="-6:5-",dy1=-14.3hs,dy2=-10.3hs>(
  \beam<dy1=-10hs,dy2=-6hs>(g*5/48 \grace<16,"/">(e g) ...
   \tuplet<format="-3-",dy1=4hs,dy2=6hs>(\beam(g*5/144 c e))
   ... ) ) \barFormat<"accolade"> \bar ... ],
 [ (* voice 2, system and staff-layout-tags removed *) ...
  \staffOff \empty*5/8 \staffOn \clef<"perc"> \space<2.5mm>
  \bar ... ], [ (* voice 3, some layout-tags removed *)
  \accol<id=2,range="3-4",style="curlyBrace">
  \barFormat<style="accolade">
  \staffFormat<style="standard",size=1.125mm> ...
  \glissando<style="wavy",dx1=2mm,dy1=1.5hs,
      dx2=-0.7mm,dy2=0hs>(  c1*1/8 \space<7mm>
  \text<"gliss."> empty*4/8 \bar<2>  \space<6mm>
  \stemsDown \marcato( { \headsRight(g3*1/8),
     f#, \headsRight(e), d# } ) )
  \staffOff ], [ (* voice 4 *) ... ] }
```

Figure 2: Example showing some of the advanced notation features of GUIDO (60% reduced in size).

two notational elements, e.g., two notes, it overrides any automatic spacing and forces the given amount of horizontal space to separate the horizontal reference positions of the elements.

Advanced GUIDO includes all tags defined in Basic GUIDO, but allows exact positioning and layout information to be specified using additional optional parameters. For instance, it is possible to exactly define the slope of a slur, or to change the size or type of a notehead, whenever this is needed. While Advanced GUIDO descriptions will usually include complete layout and formatting information, this information may also be partially specified. In this case, an application reading such a GUIDO description will try to infer unspecified information automatically where needed. Likewise, specific information not required or supported by an application can easily be ignored when interpreting the GUIDO input. Note that this approach allows the adequate representation of scores: only when exact formatting information is really needed, the corresponding additional tags and parameters have to be supplied.

Figure 1 shows a short example illustrating Advanced GUIDO's exact formatting and spacing features. The GUIDO code includes page, system, accolade, and staff formatting as well as accurate horizontal spacing information using the `\space` tag. Stem directions, beam groupings, articulations (`\ten`, `\stacc`, `\accent`) and dynamic markings (`\crescBegin`, `\crescEnd`, `\intens`) are also specified precisely.[3]

2.3 Advanced notational features

Advanced GUIDO also supports "advanced" notational features, such as glissandos, arpeggios, clusters, or figured bass indications, as well as less commonly used barline types, clefs, rehearsal marks, etc. Not all of these advanced musical concepts have standardised representations in conventional music notation; this leaves notation software supporting GUIDO some freedom in dealing with them.

While, due to the restricted space, we cannot discuss advanced notational elements in detail here, the example in Figure 2 illustrates some of them. Note, for instance, the `\staffFormat` tag in the first voice, which not only defines the staff to have three lines, but also reduces its size (compare to corresponding size-parameter in third voice). Note also the use of a percussion clef (`\clef<"perc">`) in staves 1 and 2. As can bee seen in the first voice, nested tuplets can be described in a straightforward way using `\tuplet` tags with nested ranges; nested beams are represented analogously. Voice 3 shows the usage of the `\glissando` tag, which allows not only the style but also the exact offsets from the reference positions to be specified. Furthermore, the second voice demonstrates the use of the `\staffOff` and `\staffOn` tag, to prevent sections of the staff from being shown.

3 Notation Software Supporting GUIDO

GUIDO NoteViewer & NoteServer. GUIDO NoteViewer was developed as a standalone program to display and print GUIDO files. At this time, GUIDO NoteViewer supports almost all Basic GUIDO tags. Many of the Advanced GUIDO features are currently being implemented and will be available soon. As GUIDO NoteViewer was being developed from scratch, it uses GUIDO as its internal data format. Almost all of the algorithms within GUIDO NoteViewer (like, for example, the automatic inference of `\bar` tags from the `\meter` tag) are implemented as GUIDO-to-GUIDO transformations. It is always possible to extract the internal GUIDO representation to understand how certain algorithms work.

To further promote GUIDO and its wide-spread use, we developed the GUIDO NoteServer, a publicly available notation service which can be accessed easily through the WWW (Renz, Hoos; 1998).[4] GUIDO descriptions are transmitted to GUIDO NoteServer through the CGI (common gateway interface) or through JAVA-based mechanisms. The server then creates a GIF picture of the corresponding conventional score and displays it in the user's web browser. GUIDO NoteServer also supports the embedding of dynamically created scores in arbitrary web pages. As GUIDO NoteServer renders the score from a textual, human-readable GUIDO representation, it is relatively

[3] The complete GUIDO sources for the examples presented here can be found at the GUIDO Music Notation Page, *http://www.informatik.tu-darmstadt.de/AFS/GUIDO.*

[4] The GUIDO NoteServer can be publicly accessed via *http://www.informatik.tu-darmstadt.de/AFS/GUIDO.*

easy to create and edit such embedded dynamic scores without using any additional notation software.

NoteAbility. NoteAbility is a professional music notation program developed by Keith Hamel (Hamel; 1998). Originally developed under NextStep/OpenStep, it now has versions (NoteAbility Pro) running on Mac OS-X, and OpenStep, and simplified versions (NoteAbility Lite) running on Mac OS and Windows 95/98/NT. GUIDO support has been implemented in all current versions of NoteAbility. It is possible to export complete Advanced GUIDO files (including all formatting and spacing information). As well, Basic GUIDO files can be imported, and GUIDO text fragments can be pasted directly into the score (where the code is converted into music images.) In addition to direct import and export, GUIDO is used as an intermediary stage in the importing MIDI files in NoteAbility Lite. Finally, NoteAbility Pro (running under Mac OS-X) allows fully specified Advanced GUIDO files to be imported. Like the NoteViewer/Server, NoteAbility uses the GUIDO Parser Kit as a basis for implementing GUIDO import. The portable Parser Kit, which greatly facilitates the parsing of GUIDO files, can be obtained from the authors.

4 Conclusions and Future Work

In this paper, we described Advanced GUIDO Music Notation, the second layer of GUIDO Music Notation (Hoos *et.al.*; 1998a), which is capable of encoding the complete formatting and layout of complex graphical scores. Generally, while the GUIDO design focusses on purely musical and logical concepts, it also covers graphical and performance-related aspects of music representation. Based on a conceptually simple yet powerful syntax, GUIDO is a human-readable and portable text format. The most important feature of the GUIDO approach is *representational adequacy*, meaning that simple musical concepts can be expressed in a syntactically simple way, and only musically more complex material may require more complex representations.

GUIDO differs in many important aspects from existing approaches for music representation like DARMS, *cmn*, NIFF, SMDL, or MIDI (Selfridge 1997; Sloan; 1997; Grande, Belkin; 1996). These differences are mainly with respect to scope and representational adequacy. Unlike NIFF, SMDL, or *cmn*, GUIDO is not primarily targeted or restricted to music notation applications. Rather it aims, like DARMS, at covering a broad range of applications, including composition software, analytical tools, and general computer music environments, such as SALIERI (Hoos *et.al.*; 1998b) or Open Music (Assayag *et.al.*; 1997). As we have demonstrated in this paper, Advanced GUIDO can represent all the information required for exact score formatting. Thus, like *cmn*, SMDL, or NIFF, GUIDO can be used as a notation interchange format. However, since GUIDO encodes musical and graphical information in a content-oriented, human-readable, representationally adequate way, it is considerably easier to implement. Furthermore GUIDO can be easily and naturally extended to accommodate the requirements of specific applications. As in the case of HTML or XML, this is achieved in such a way that applications which do not recognise a specific GUIDO extension can still utilise all the remaining information in a GUIDO description.

The current development of GUIDO NoteViewer/Server is directed towards providing complete support of the Advanced GUIDO specification. This is closely related with research on algorithms for notation and their descriptions as GUIDO-to-GUIDO transformations. Additionally, we have started implementing a platform-independent GUIDO Renderer that will further facilitate the use of GUIDO for music notation on various platforms and within different applications. The GUIDO Renderer uses the newly developed GUIDO Graphic Stream protocol for the communication between a platform-independent notation engine and front-ends for different platforms. We plan to finish implementing a prototype of the GUIDO Renderer by the end of 1999.

First versions of converters between GUIDO and MIDI as well as Csound score files have been implemented and other converters are planned. Furthermore, conversions between GUIDO and other formats, such as MAX Q-Lists, can be done via NoteAbility. We are currently extending the GUIDO support for the SALIERI computer music system (Hoos *et.al.*; 1998b); this will provide an easy, yet powerful way for algorithmically manipulating GUIDO descriptions, and thus facilitate compositional and analytical applications of GUIDO. Finally, we plan to extend GUIDO with watermarking techniques and to embed GUIDO into an XML DTD. We see a great potential for GUIDO as a powerful and representationally adequate music representation language and notation interchange format, and hope to see many musical applications supporting and benefiting from GUIDO.

References

Gérard Assayag, Carlos Agon, Joshua Fineberg, Peter Hanappe; An Object Oriented Visual Environment For Musical Composition; *in:* Proc. of ICMC'97

Cindy Grande, Alan Belkin; The Development of the Notation Interchange Format; Computer Music Journal 20(4):33-43; MIT 1996

Keith A. Hamel; NoteAbility - A Comprehensive Music Notation Editor; *in:* Proc. of ICMC'98

Holger H. Hoos, Keith A. Hamel, Kai Renz, Jürgen Kilian; The GUIDO Notation Format - A Novel Approach for Adequatly Representing Score-Level Music; *in:* Proc. of ICMC'98

Holger Hoos, Jürgen Kilian, Kai Renz, Thomas Helbich; SALIERI – A general, interactive Computer Music System; *in:* Proc. of ICMC'98

Holger H. Hoos, Keith A. Hamel; The GUIDO Music Notation Format – Specification Part 1; Technical Report TI 20/97; Darmstadt University of Technology; Darmstadt 1997; available from `http://www.informatik.tu-darmstadt.de/AFS/GUIDO`

Kai Renz, Holger H. Hoos; A WEB based Approach to Music Notation using GUIDO; *in:* Proc. of ICMC'98

E. Selfridge-Field (ed.); Beyond MIDI, The Handbook of Musical Codes; MIT Press; Cambridge, Massachusetts 1997

Donald Sloan; HyTime and Standard Music Description Language: A Document-Description Approach; *in:* Beyond MIDI, The Handbook of Musical Codes; Eleanor Selfridge-Field (ed.); pp. 469-490; MIT Press; Cambridge, Massachusetts 1997