

5 Representing Score-Level Music Using the GUIDO Music-Notation Format

Holger Hoos¹, Keith Hamel², Kai Renz³, and Jürgen Kilian³

¹Department of Computer Science
University of British Columbia
2366 Main Mall
Vancouver, BC, V6T 1Z4, Canada
hoos@cs.ubc.ca

³Department of Computer Science
Technical University, Darmstadt
Wilhelminenstr. 7
64283 Darmstadt, Germany
renz@iti.informatik.tu-darmstadt.de
kilian@iti.informatik.tu-darmstadt.de

²School of Music
University of British Columbia
6361 Memorial Road
Vancouver, BC, V6T 1Z2, Canada
hamel@interchange.ubc.ca

Abstract

GUIDO Music Notation is a novel approach for adequately representing score-level music. Based on a simple, yet powerful and easily extensible formalism, GUIDO is realized as a plain-text, human-readable and platform-independent format. The key feature of the underlying design is *representational adequacy*: simple musical concepts can be expressed in a simple way, while complex musical notions may require more complex representations. GUIDO Music Notation can be used for a broad range of applications, including notation software, composition and analysis systems and tools, music databases, and music on the World Wide Web. In this article, we discuss the motivation for developing GUIDO Music Notation, give an overview of its design and features, and describe its current applications.

5.1 Introduction and Background

¹ GUIDO Music Notation is named after Guido d'Arezzo (ca. 992-1050), a renowned music theorist of his time and important contributor to today's conventional musical notation. His achievements include the perfection of the staff system for music notation and the invention of solmization (*solfege*)—a scheme for assigning syllabic names (e.g. *do, re, mi*) to pitch positions within the scale, mode, or hexachord.

The GUIDO Music-Notation Format¹ is a general-purpose, formal language for representing score-level music in a platform-independent, plain-text and human-readable way. The GUIDO design concentrates on general musical concepts (as opposed to only notational, i.e. graphical features). Its key feature is *representational adequacy*, meaning that simple musical concepts should be represented in a simple way and only complex notions should require complex representations. One important consequence of this design objective is GUIDO's ability to represent incomplete musical information, such as a scale without duration or octave information.

The GUIDO design is organized in three layers—Basic, Advanced, and Extended GUIDO Music Notation. *Basic GUIDO* introduces the fundamental GUIDO syntactical structures and covers basic musical notions; *Advanced GUIDO* extends this layer to support exact score formatting and more sophisticated musical concepts; and *Extended GUIDO* introduces features which are beyond conventional music notation, such as absolute timing, microtonal tuning, and hierarchical score representations.

GUIDO Music Notation is designed as a flexible and easily extensible open standard. In particular, its syntax does not restrict the features which can be represented. Thus, GUIDO can be easily adapted and customized to include specialized musical concepts that might be required in the context of research projects in computational musicology. More important, GUIDO is designed so that when custom extensions are used, the resulting GUIDO data can still be processed by other applications that support GUIDO but are not aware of the custom extensions, which are gracefully ignored. This design facilitates the incremental implementation of GUIDO support in music software, which, especially for research software and prototypes, can speed up the software development process significantly.

GUIDO has not been developed with a particular type of application in mind but to provide an adequate representation formalism for score-level music over a broad range of applications. The intended application areas include notation software, compositional and analytical systems and tools, musical databases, performance systems, and music on the World Wide Web.

When comparing GUIDO to other music representation formalisms, several important differences can be found. Unlike the MIDI File Format (Hewlett, Selfridge-Field 1997) or NIFF (Grande, Belkin 1996), GUIDO is a plain-text, human-readable format. This facilitates the implementation of GUIDO support in software applications and is highly advantageous when GUIDO is used to interchange musical data between applications, across platforms, or over the internet. Compared to other plain-text approaches like DARMS (Selfridge-Field 1997), SMDL (Sloan 1997), or music representations based on HTML or XML, GUIDO exhibits better representational adequacy (in the sense defined above) which again facilitates both its implementation and its use. Finally, in contrast to representations which are specifically designed for graphic music notation, like *Common Music Notation (cmn)* (Schottstaedt 1997), the GUIDO design is focused on music information, while also supporting exact score-formatting features. For these reasons, we believe that GUIDO is equally well-suited for score setting and other music applications. In this respect, GUIDO follows a similar goal to DARMS, which was used for a wide range of music applications until the early 1980s.

Figure 1. A simple scale represented in GUIDO, DARMS, and *cmn*. (Note that the clef is explicitly represented, while barline and meter information are not.)



GUIDO:
`[\clef<"treble"> d1/4 e/8 f# g a b c#2 d/2 _/2]`

DARMS:
`||1 !G 0Q 1E 2# 3 4 5 6#H RH`

cmn:
`(cmn staff treble d4 q e4 e fs4 e g4 e a4 e b4 e cs5 h half-rest)`

5.2 Basic GUIDO Music Notation

Generally, GUIDO Music Notation is designed for the adequate representation of musical material, which can then be interpreted by computer music software or used for conventional music notation. The Basic GUIDO layer covers the relevant aspects for representing “simple” music, from primitive musical objects such as single notes or rests, to complete pieces with multiple voices, dynamics, and other markings, as well as text. There is no inherent restriction on the length or complexity

of pieces specified in Basic GUIDO (although applications supporting GUIDO might impose such restrictions). This layer of GUIDO focuses on representing basic musical concepts; it does not, for instance, cover the exact placement or the graphical appearance of musical objects (which can be specified in the next layer, Advanced GUIDO).

GUIDO Music Notation is based on two basic syntactic elements: events and tags. An *event* is a musical entity which has a duration (e.g. notes and rests) whereas *tags* are used to define musical attributes (such as meter, clef, and key indications). All layers of GUIDO are based on the same simple syntax which is defined below.

5.2.1 Notes and Rests

In GUIDO notes are represented by their names (e.g. c, d, e or do, re, mi) followed by optional additional parameters: accidentals, register, and duration. By leaving out these parameters, incompletely specified music can be represented in GUIDO. A complete note description in GUIDO looks like `c#1*1/4`—which represents the C# above middle C with a duration of a quarter note². Note that register and duration information is carried over from one note to the next within note sequences. As GUIDO has been developed for a broad range of musical applications, different systems of diatonic note names (e.g. *do*, *re*, *mi*) are supported. In addition to the commonly used pitch-class names, GUIDO also provides chromatic pitch-classes. There is, for example, a pitch-class called *cis* that is different from *c#* and *d&* (where the # denotes a sharp and the & denotes a flat), even though when played on a regular MIDI device the three notes sound the same. This concept is important for instances (such as in serial music) where pitch classes are better represented without accidental indications. Rests are represented like notes, but instead of a note name, an underscore is used: `_*1/4`. Alternatively, the special note name *rest* can be used. Obviously, rests don't have accidentals or octave information; rest durations are specified in the same format as that of notes.

²The * between the register and the duration is needed to distinguish between register and duration information.

5.2.2 Sequences and Segments

GUIDO provides two orthogonal constructs for grouping notes and rests into larger structures: sequences and segments. A *sequence* describes a series of temporally consecutive musical objects, whereas a *segment*

describes a number of simultaneous musical objects. The most simple form of a segment is a chord where the voices consist of single notes.

These two concepts have not only been chosen for reasons of formal simplicity and elegance, they are also motivated by the fact that many musical pieces can be represented in two fundamentally different ways: as a sequence of chords (chord-normalform) or as a set of simultaneous voices (poly-normalform). In the former case, the music is described as a sequence of chords (which is a sequence of segments), while in the latter case, it is represented by set of sequences (a segment of sequences). The use of either one of these normalforms for the description of a piece depends largely on the style of the piece: homophonic choral music can often very naturally be written in chord-normalform, whereas a contrapuntal composition is typically more adequately described using poly-normalform. Basic GUIDO facilitates both normalforms by supporting an extended poly-normalform which allows chords within sequences of a segment (i.e. chords within polyphonic voices). Extended GUIDO allows more complex nestings of sequence and segment constructors (refer to the section on Extended GUIDO below).

Sequences and segments are realized in GUIDO using the following syntax: a sequence begins with a square bracket ([) and ends with the corresponding closing bracket (]). Segments are enclosed in curly braces ({...}). Musical objects within segments are separated by commas. Within sequences and chords, octaves and durations are implied from the previous note, if they are not explicitly specified. It should be noted that GUIDO is a free-text format, i.e. the position of line-breaks and the formatting of the textual GUIDO data do not affect its interpretation. GUIDO also allows comments and annotations, which are ignored when interpreting the GUIDO data (see Figure 6).

5.2.3 Tags and Basic Musical Concepts

Music does not only consist of notes and rests, but also contains additional musical and graphical information. GUIDO can represent all the commonly known musical attributes using *tags*: a tag has a name, optional parameters, and an optional range of application. A large number of tags are defined in Basic GUIDO; for a complete description see Hoos, Hamel (1997). For example, there are tags to describe clefs, meter, tempo, dynamics, crescendo, accelerando, and so on. The

semantics of the tags defined in GUIDO can normally be inferred from their names. Figure 3 illustrates the use of tags in Basic GUIDO.

Figure 2. Examples of Basic GUIDO: a) single notes and rests, b) single voices and multi-voice pieces, c) chord sequence vs. poly-normalform. Note that in these examples, only notes and rests are represented in the GUIDO data.

a)



OR

```

c2*1/8 d#-1*1/2.. rest*1/4. b&3*2/1 g##-2*1/10 e&&1*1/24
c2/8 d#-1/2.. /4. b&3*2 g##-2/10 e&&1/24

```

b)

Schumann Op68 #1



```
[ e2/4 d c b1 a/8 c2 b1 d2 c/4 g1 ]
```

Schumann Op68 #1



```
{ [ e2/4 d c b1 a/8 c2 b1 d2 c/4 g1 ],
  [ c1/8 g f g e g c e f d g f e d ] }
```

c)


```
[ {d2/4 ,f#1 ,d1 }{a1 ,f# ,d }{b ,g ,d }
  {a ,f# ,d }{g/2 ,e ,d }{a ,e ,c# } ]
```

OR

```
{ [ d2/4 a1 b a g/2 a ] ,
  [ f#1/4 f# g f# e/2 e ] ,
  [ d1/4 d d d d/2 c# ] }
```

Figure 3. Basic GUIDO with tags (Mozart K. 576).

Adagio



```

[ [ \tempo<"Adagio"> \staff<1>
  \clef<"g2"> \key<"A"> \meter<"3/4">
  \intens<"fp">
  \beam(c#2*1/8. d/64 c# b1 c#2)
  \beam(e/8 d b1 g#) \bar
  a. c#2/32 a1 \tie(e2/2 e/16) ... ] ,
  [ \staff<2> \clef<"bass">
  \tie(e1*1/2. \bar e1*1/4) ... ] ,
  [ \staff<2>
  a0*1/4 b d1 \bar c# \clef<"treble">
  _1/8 \stacc( { d2*1/8, g#1*1/8 }
  { d2*1/8, g#1 } { d2*1/8, g#1 } ... ] ]

```

5.3 Advanced GUIDO Music Notation

Advanced GUIDO comprises some of the more advanced concepts not covered in Basic GUIDO, such as glissandos, arpeggios, clusters, different types of noteheads, different types of staves, and many features from contemporary music notation. It also addresses issues of score formatting such as exact spacing and positioning of all music and graphic elements. Advanced GUIDO is based on the same syntax as Basic GUIDO with three minor extensions: named tag parameters, parameter values with units, and multiple overlapping tag ranges.

By using additional optional parameters, Advanced GUIDO extends Basic GUIDO tags to include precise information about the graphical appearance of the object as well as exact positioning and layout information. For example, it is possible to define the midpoint of a slur, the height and angle of a beam, or the size and type of a notehead. In Advanced GUIDO, tag parameters are normally specified by using pre-defined names (e.g. `\clef<type="g2", size=0.5>` which specifies a g-clef on the second line—a treble clef—which is 50% of its standard size). Parameter names are optional, however, and they can be omitted provided that all parameters are listed (e.g. `\clef<"g2", 0.5>`). The Advanced GUIDO specification allows a variety of measurement units to be used in conjunction with tag parameters. These units (e.g. `cm`, `mm`, `in` [inches], `pt` [points], `pc` [picas], and `hs` [halfspaces]) can be placed directly after the numerical values of all tag parameters which refer to size or position. For example, the tag `\pageFormat<20cm, 40cm>` defines a score page to be 20 centimeters wide by 40 centimeters high.

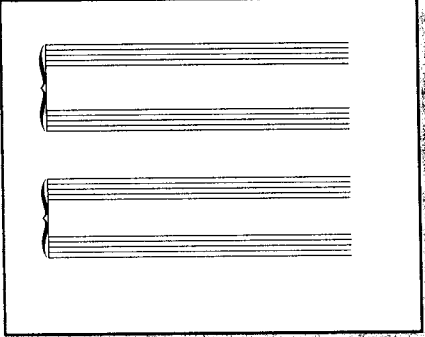
With Advanced GUIDO, it is possible to specify exact score-formatting information that could be used by any kind of professional notation software. This feature makes Advanced GUIDO an ideal candidate for a platform- and application-independent notation interchange format. Support for score formatting and layout details is accomplished by adding a few new tags and by adding a large number of tag parameters to existing Basic GUIDO tags. The new tags are used to specify the physical dimensions and margins of score pages (`\pageFormat`), the relative positioning of systems on a page (`\systemFormat`), the type and size of staves in a system (`\staffFormat`), their relative location (`\staff`), and page and system breaks (`\newPage`, `newSystem`). While some elements (such as systems or a score title) are positioned relative to the page, most are positioned relative to the staves or notes they are logically associated with. Generally, for elements associated with a staff, vertical offsets are

³ A halfspace is half the vertical distance between two staff lines.

specified relative to the size of the staff, using the unit halfspace (hs).³ Similarly, the size of many elements is specified relative to the size of the staff they appear on (i.e. `<size=0.75>` would indicate that an object is 3/4 of its normal size for the staff it is on). The most important mechanism for indicating exact horizontal locations is the `\space` tag. When placed between two elements, the `\space` tag forces the given amount of horizontal space to separate the reference positions of the two elements. The horizontal distance can be indicated in any of the available measurement units (e.g. `\space<2.57mm>` or `\space<16.9pt>`).


Figure 4. Exact formatting and layout in Advanced GUIDO: a) empty page with exact positioning and formatting of systems and staves; b) exact specification of vertical positions; c) precise horizontal spacing of notes and graphical elements. [Figure shown at 55% of original size.]

a)




```
[ [ \pageFormat<w=10cm,h=8cm,lm=12mm,tm=11mm,
    rm=17mm,bm=23.5mm>
  \systemFormat<staves="1-2",dx=0mm>
  \accol<id=0,range="1-2",style="curlyBrace">
  \staff<id=1,dy=14mm>
  \staffFormat<style="standard",size=1mm> ...
  \newSystem<dy=3cm>
  \systemFormat<staves="1-2",dx=0mm>
  \accol<id=0,range="1-2",style="curlyBrace">
  \staff<id=1,dy=14mm>
  \staffFormat<style="standard",size=1mm> ... ] ,
  [ \staff<id=2>
  \staffFormat<style="standard",size=1mm> ... ] ]
```

b)



```
[ ...
  \intens<"f",dy=<23mm>
  \beam<dy=-7.00hs,dy2=-9.00hs> (g1*1/8 b e2)
  ...
  \intens<"f",dy=<14mm>
  \beam<dy=-12.00hs,dy2=7.00hs> (g1*1/8 b e2)
  ... ]
```

c)



```
[ ... \space<2.5mm> \clef<"g2"> \space<3mm>
  \meter<"4/4"> \space<4mm> b1*1/2
  \space<13mm> c#*1/4 \space<6.5mm> d2*1/4
  \space<6mm> \bar ... ]

[ ... \space<2.5mm> \clef<"g2"> \space<4.5mm>
  \meter<"4/4"> \space<4.5mm> b1*1/2
  \acc<dx=-2.5mm>(\space<7.5mm> c#*1/4
  \space<6.5mm> d2*1/4 \space<6mm> \bar ... ]
```


Although Basic GUIDO supports overlapping tag ranges by using begin/end tags as in `\slur(c1/2 \crescBegin e/8 d) g/4 \crescEnd`, it does not handle the rare cases where multiple occurrences of the same tag have overlapping ranges. In order to support overlapping tag ranges, Advanced GUIDO allows explicit disambiguation of begin and end tags. For every tag `\id` which can be used with a range, the tagged range `\id<...>(...)` can also be represented as `\idBegin<...>... \idEnd`. Both the `\idBegin` and `\idEnd` tags are used without ranges, and the `\idBegin` tag takes all the tag parameters. To accommodate overlapping ranges, the begin/end tags can be extended with a unique numeric postfix, using the syntax `\idBegin:n<...>... \idEnd:n`. Thus, three overlapping slurs can be represented as `\slurBegin:1... \slurBegin:2... \slurEnd:1... \slurBegin:3... \slurEnd:2... \slurEnd:3`.

Although Advanced GUIDO descriptions will usually include complete layout and formatting information, this information may also be partially specified. An application reading a partially specified GUIDO description should try to infer unspecified information when needed. Likewise, specific information not required or supported by an application can easily be ignored when interpreting the GUIDO input. This approach allows the adequate representation of scores: only in situations where exact formatting information is really needed do the additional tags and parameters have to be supplied or interpreted.

Figure 5 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 precise horizontal spacing information using the `\space` tag. Stem directions, beam groupings, articulations (`\ten`, `\stacc`, `\accent`), and dynamic markings (`\crescBegin`, `\crescEnd`, `\intens`) are also included. (N.B. The complete GUIDO source for the examples presented here can be found at the GUIDO Music Notation Page www.salieri.org/GUIDO.)

Advanced GUIDO also supports “advanced” notational features such as glissandos, arpeggios, note clusters, rehearsal marks, and figured bass indications, as well as less commonly used barline types, clefs, etc. Not all of these musical concepts have standardized representations in conventional music notation; this leaves the notation software supporting GUIDO the freedom to deal with them or to ignore them. Figure 6 illustrates some of the non-standard features supported by Advanced

Figure 5. Advanced GUIDO (Scriabin Op. 8, No. 2): note how the GUIDO data represents all the layout and spacing information for the various notational elements and symbols. [Figure shown at 50% of original size.]



```
(\pageFormat<w=151mm,h=151mm,lm=14mm,tm=25mm,m=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,dc2=1mm,dy2=11.5mm,h=4mm>
\beam<dy1=11hs,dy2=9hs>(\ten(#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<mf="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 ...} ]
```

GUIDO. Notice that the `\staffFormat` tag in the first voice not only defines the staff to have three lines, it also reduces its size. Note also the use of a percussion clef (`\clef<"perc">`) in staves 1 and 2, and the representation of nested triplets (using `\tuplet` tags with nested ranges). Voice 3 shows the use of the `\glissando` tag, which allows both the style and the exact offsets from the reference positions to be specified. Furthermore, voice 2 demonstrates the use of the `\staffOff` and `\staffOn` tags which indicate that a section of the staff is not visible.

5.4 Extended GUIDO Music Notation

The third level of the GUIDO design includes various features which extend Basic and Advanced GUIDO beyond conventional music notation. Unlike Basic and Advanced GUIDO, the Extended GUIDO specification is work in progress and—in its final version—will reflect the experiences and needs of the groups involved in using and testing

GUIDO to date. Extended GUIDO is based on minor and very natural extensions of the fundamental GUIDO syntax and an extended set of tags and events. Here, we give a brief overview of some of the features covered by Extended GUIDO.

Figure 6. Non-standard notational features in Advanced GUIDO. The text enclosed in (*...*) represents comments which are ignored when interpreting the GUIDO data. [Figure shown at 60% of original size.]

```
{ [ (* 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>(b*5/48 \grace<16,"/">(g b) ...
  \tuplet<format="-3-",dy1=4hs,dy2=6hs>(\beam(b*5/144 e g)) ... )
  \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( { g3*1/8,
  \headsLeft(ff#), e, \headsLeft(d#) } ) ) \staffOff ] ,
[ (* voice 4 *) ... ] }
```

5.4.1 Microtonality and Tuning

While certain aspects of microtonality can already be realized in Basic GUIDO, such as the differences between $d\sharp$, e , or $f\flat$, Extended GUIDO also introduces representations for other types of microtonal information, such that different notions of microtonality, such as just tuning, microtonal alterations, arbitrary scales, or non-standard tuning systems can be adequately represented. For example, using the new `\alter` tag, microtonal alteration of standard pitch classes, such as quarter-tone accidentals (see Figure 7a) can be realized; `\alter<-0.5>(a1/8)` represents an eighth note $a1$ altered downwards by a quarter tone. Likewise, different tuning systems can be represented by two other new tags, `\tuning` and `\tuningMap`, which are used to select predefined tuning systems, such as just tuning, or to specify a tuning scheme based on systematic alterations of given pitch classes. Finally, concepts such as quarter-tone scales or absolute pitches can be represented by parameterized events, which generalize the concept of an event (such as a note or rest) in Basic or Advanced GUIDO.

5.4.2 Exact Timing

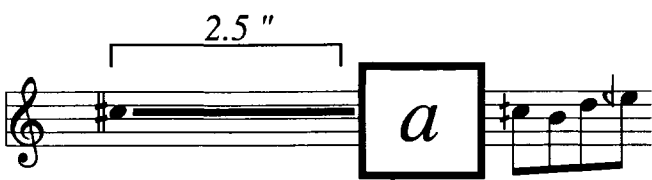
In Basic and Advanced GUIDO, note and rest durations are always specified as relative durations. Extended GUIDO also allows exact timing by specifying absolute durations, as exemplified in Figure 7a, where `c#2*2500ms` represents the note $c\sharp 2$ with a duration of exactly 2500 milliseconds. As also shown in Figure 7a, relative and absolute durations can be freely combined in a given fragment or piece.

5.4.3 Hierarchical and Abstract Scores

Extended GUIDO facilitates the direct representation of *hierarchical score structures* by allowing arbitrary nesting of the sequence and segment constructors used in Basic and Advanced GUIDO. A simple example for this is shown in Figure 7b; note that while there is no standard graphical representation of such hierarchical scores, they can be extremely useful in analytical applications. Similarly, Extended GUIDO supports the representation of *abstract scores*, which are basically scores containing variables or “holes” (see Figure 7a). These can be useful for representing incomplete structural information such as musical schemata.

Figure 7. Some features of Extended GUIDO: a) absolute note durations, variables (abstract scores), and microtonal alteration; b) a simple example of a hierarchical score.

a)



{ [c#2*2500ms ref<a> \alter<+0.5> (c2/8) b1 d2 \alter<-0.5> (e)] }

b)



{ [c2/8 d b1 e2 _/2
 { [a1\8 e2 d b1 a1 e2 d c2] , [c2/8 d b1 e2 c d b1 e2]
 c2/8 d b1 e2 _/2] }

5.5 Applications of GUIDO

GUIDO Music Notation has been designed as a music representation format to be used by a broad range of applications. It is intended to be used both as an internal storage format as well as an interchange format facilitating the access of the same musical information from different software applications or tools. In the following section, we give a brief overview of applications and projects that are currently using GUIDO.

5.5.1 Notation Software

NoteAbility is a professional music notation program developed by Keith Hamel. 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 (7.5+) and Windows 95/98/NT. GUIDO support has been implemented in all

versions. Currently, Advanced GUIDO data can be exported and Advanced GUIDO import is being implemented. Furthermore, *NoteAbility Lite* uses GUIDO as the intermediary stage in importing MIDI files. The goal of the current development is to implement full Advanced GUIDO support, so that GUIDO can be used for exchanging score data between the different versions of *NoteAbility*, as well as between *NoteAbility* and other music software environments.

GUIDO NoteViewer is a stand-alone program for displaying and printing GUIDO files. The current version supports most of Basic GUIDO, and Advanced GUIDO support is under development and will be available soon. GUIDO NoteViewer is based on a platform-independent notation renderer module (the GUIDO Notation Engine) and a platform-dependent graphical user interface. This design allows GUIDO NoteViewer to be easily ported to different hardware platforms and operating systems. Based on the GUIDO Notation Engine, we developed a publicly available World Wide Web server, the GUIDO NoteServer (Renz, Hoos 1998), which transforms textual GUIDO descriptions into graphical representations of the corresponding score, which can then be downloaded. This service can be used from any computer with internet access, and is particularly useful for embedding music in Web pages. The advantage of the GUIDO NoteServer lies in the fact that the rendering uses the textual GUIDO representation. To change the music notation returned by the server, it is only necessary to change the textual representation; no additional notation software is needed.

5.5.2 Music Databases

Music databases play an important role, especially in the context of online systems available on the World Wide Web. As these databases grow in number, size, and complexity, it becomes increasingly important to provide flexible and efficient search and retrieval techniques. By using GUIDO Music Notation instead of other representation languages (in particular MIDI), the musical data stored in the database, as well as the queries used to access them, can include all the relevant information from the original score or fragment in an adequate and natural way.

Recently, we developed a prototypical music database and retrieval system (GUIDO/MIR) as a Web-based system built on GUIDO Music Notation and the simple and intuitive “Query By Example” (QBE) approach for

the retrieval of music data (Görg 1998). Since GUIDO supports a representationally adequate description of music data, queries can be as simple as a melodic contour (given as a sequence of note names), or can include additional music information (such as durations, slurs, or metrical structure). Our matching algorithm supports different degrees of precision for the matching process for pitch and duration, from absolute precision to very high degrees of tolerance. A specialized but simple extension of the GUIDO language for queries provides fine-grain control over the matching precision on the level of individual notes. Our experimental database contains several hundred pieces of varying complexity, which were all automatically converted from various other formats into GUIDO. The prototype of the GUIDO/MIR Server can be accessed via the internet using any standard WWW browser.

5.5.3 Computer Music Systems and Tools

The origins of GUIDO are closely connected to our own efforts to build a general, interactive computer music environment, the SALIERI System (Hoos et al. 1998b). An integral part of this software system is the SALIERI Language, a music programming language which uses GUIDO Music Notation as its fundamental music representation formalism. The SALIERI System serves as a platform for computer-aided composition, pre-compositional design, and music analysis.

As part of a cooperation with GRAME in Lyon, France, we are developing GUIDO support and customized GUIDO extensions for Elody, a compositional system based on lambda calculus (Orlarey et al. 1997). Other projects involve the realization of GUIDO support for HARMONET, a neural-network-based harmonizer developed at the University of Karlsruhe, Germany (Hild et al. 1992), and Mutabor, a microtonal instrument developed at the Universities of Darmstadt and Dresden, Germany (Abel et al. 1992).

5.5.4 Converting Between GUIDO and Other Formats

Given the multitude of electronic musical data and music software packages, it is essential that one can convert between as many of these representations as possible. In particular, it seems crucial to be able to

convert between GUIDO and MIDI, since MIDI is so widely used as a playback and storage format for score-level music.

For MIDI playback and for data exchange from GUIDO to MIDI applications (e.g. sequencer software), a GUIDO-to-MIDI converter (GMN2MIDI) was built. The pitch and timing information of GUIDO files can be easily represented in MIDI files. However, since a MIDI file does not contain any structural or graphical information, much of the original data is lost in the conversion. In the future, this converter could be extended to support variants of MIDI (such as MIDI+) which are capable of representing additional score-level information (Hewlett, Selfridge-Field 1997). The conversion of MIDI into Extended GUIDO files could be easily done by using exact timing information and mapping the controller and meta-events of the MIDI file to user-defined Extended GUIDO tags (see also section on Extended GUIDO).

The conversion of live performed MIDI files into Basic GUIDO descriptions is a far more difficult task because a considerable amount of high-level information, such as relative note durations, slurs, and dynamic marks, has to be inferred from the low-level data in the MIDI file. HEISENBERG is an adequate and interactive approach for inferring score-level descriptions from low-level musical data in situations where no score-timing information (i.e. tempo or meter) is available. Because a fully automated procedure does not seem to give optimal results over a wide range of input data, HEISENBERG is designed and implemented as an interactive algorithm. The current prototype converts MIDI data into Basic GUIDO. It works in stages, including voice separation and chord detection, inference of unquantized information (such as slurs, ornaments, or glissando), tempo and meter detection, quantization of onset times and note durations, and inference of quantized information (e.g. staccato, tenuto, dynamics). In each of these stages, HEISENBERG may prompt for user input if difficulties are encountered while processing the given data.

In addition to the translators between GUIDO and MIDI data, a number of converters for other formats have been recently implemented. In particular, a GUIDO-to-*Csound* converter as well as translators from ABC (a format predominantly used in melodic databases) and *Capella* (a music notation program widely used in Europe) to GUIDO are now available.

5.5.5 GUIDO Support

GUIDO support for existing or new applications is facilitated by a portable and modular GUIDO Parser Kit, which can be easily embedded in any application to provide the basic code for reading GUIDO files. The parser does not provide any sophisticated data structure for storing the data read from GUIDO files. Instead, for each syntactic element of GUIDO (notes, rests, tags), the information about this element is passed to a function provided by the application, which can then process or store the information in an application-specific way. This approach has been chosen because the requirements of different types of GUIDO applications would make a unified data structure extremely complex and cumbersome and would thus contradict GUIDO's spirit of adequateness. The GUIDO Parser Kit has been utilized by most of the applications currently supporting GUIDO import: *NoteAbility*, the GUIDO NoteViewer, the GMN2MIDI converter, etc.

Typically, GUIDO export is easy to realize since the human-readable and representationally adequate design of GUIDO allows the user to use standard text-output routines which are available in all programming languages in a simple and straightforward way. At the same time, these features greatly facilitate the incremental implementation, testing, and debugging of GUIDO export routines.

5.6 Conclusions and Future Work

In this paper, we described the GUIDO Music Notation format as a novel way to adequately represent score-level music. Based on a conceptually simple, yet powerful formalism, GUIDO is a human-readable and portable text format which is easily extensible and can be used in a wide variety of music applications. The most important feature of the GUIDO approach is *representational adequacy*, which means that simple musical concepts can be expressed in a syntactically simple way, and only more musically complex material may require more complex representations. GUIDO Music Notation is realized as a three-level hierarchy. While Basic GUIDO covers basic musical concepts, Advanced GUIDO comprises exact score formatting and advanced musical concepts. Extended GUIDO introduces concepts which are not covered in conventional music notation such as exact timing, microtuning, generic pitch classes, hierarchical and abstract scores, and user-defined GUIDO tags and events. While the

three GUIDO levels discussed in this paper will be sufficient for most purposes, some applications will require additional concepts. Because GUIDO can easily be extended by defining additional tags and events without losing its conceptual coherence and syntactic simplicity, specialized GUIDO extensions can be used in these situations. A major advantage of using specialized GUIDO extensions is that applications supporting standard GUIDO will still be able to process the customized GUIDO data, since GUIDO applications can just ignore tags and events they cannot handle.

GUIDO differs in many important aspects from existing approaches for music representation like DARMS, *cmn*, NIFF, SMDL, or MIDI. These differences are mainly with respect to the scope and representational adequacy of the representations. Unlike NIFF, SMDL, or *cmn*, GUIDO is not primarily targeted toward nor restricted to music-notation applications. Rather it aims, like DARMS, at covering a broad range of music 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 was 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 recognize a specific GUIDO extension can still utilize all the remaining information in a GUIDO description.

From our experience so far, we believe that GUIDO Music Notation will prove to be extremely useful both as a music representation format and as a data exchange format for music information. It is our hope that by presenting the concepts underlying GUIDO and by encouraging the use of GUIDO by members of the music community working in a wide range of activities, GUIDO Music Notation will ultimately become a widely used formalism for representing score-level music.

References

- Abel, Volker, Peter Reiss, and Rudolf Wille (1992). "MUTABOR II: ein computergesteuertes Musikinstrument zum Experimentieren mit Stimmungslagiken und Mikrotönen." Technische Hochschule Darmstadt, Fachbereich Mathematik, Preprint Nr. 1513.
- Assayag, Gerard, Carlos Agon, Joshua Fineberg, Peter Hanappe (1997). "An Object-Oriented Visual Environment for Musical Composition" in *Proceedings of the International Computer Music Conference 1997* (San Francisco: International Computer Music Association), pp. 364–367.
- Görg, Marco (1998). "GUIDO/MIR: ein GUIDO-basiertes Music Information Retrieval System." M.Sc. thesis. Technische Universität Darmstadt, Fachbereich Informatik.
- Grande, Cindy, and Alan Belkin (1996). "The Development of the Notation Interchange Format," *Computer Music Journal* 20/4, 33–43.
- Hewlett, Walter B., Eleanor Selfridge-Field, et al. (1997). "MIDI" in *Beyond MIDI: The Handbook of Musical Codes*, ed. Eleanor Selfridge-Field (Cambridge, MA: MIT Press), pp. 41–72.
- Hild, Hermann, Johannes Feulner, and Wolfram Menzel (1992). "HARMONET: A Neural Net for Harmonizing Chorales in the Style of J. S. Bach" in *Advances in Neural Information Processing Systems 4 (NIPS*4)* (San Mateo, CA: Morgan Kaufmann Publishers), pp. 267–274.
- Hoos, Holger H., and Keith A. Hamel (1997). "The GUIDO Music Notation Format-Specification, Part 1." Technische Universität Darmstadt, Technical Report TI 20/97. [Available from www.informatik.tu-darmstadt.de/AFS/GUIDO]
- Hoos, Holger H., Keith A. Hamel, Kai Renz, and Jürgen Kilian (1998). "The GUIDO Notation Format—A Novel Approach for Adequately Representing Score-Level Music" in *Proceedings of the International Computer Music Conference 1998* (San Francisco: International Computer Music Association), pp. 451–454.
- Hoos, Holger H., Jürgen Kilian, Kai Renz, and Thomas Helbich (1998). "Salieri—A General, Interactive Computer Music System" in *Proceedings of the International Computer Music Conference 1998* (San Francisco: International Computer Music Association), pp. 385–392.
- Orlarey, Yann, Dominique Fober, and Stéphane Letz (1997). "L'environnement de composition musicale Elody" in *Proceedings of Journées d'Informatique Musicale 1997* (Lyon: Grame—Centre National de Création Musicale), pp. 122–136.

- Renz, Kai, and Holger H. Hoos (1998). "A Web-Based Approach to Music Notation Using GUIDO" in *Proceedings of the International Computer Music Conference 1998* (San Francisco: International Computer Music Association), pp. 455–458.
- Schottstaedt, Bill (1997). "Common Music Notation" in *Beyond MIDI: The Handbook of Musical Codes*, ed. Eleanor Selfridge-Field (Cambridge, MA: MIT Press), pp. 217–221.
- Selfridge-Field, Eleanor (1997). "DARMS, Its Dialects, and Its Uses" in *Beyond MIDI: The Handbook of Musical Codes*, ed. Eleanor Selfridge-Field (Cambridge, MA: MIT Press), pp. 163–174.
- Sloan, Donald (1997). "HyTime and Standard Music Description Language: A Document-Description Approach" in *Beyond MIDI: The Handbook of Musical Codes*, ed. Eleanor Selfridge-Field (Cambridge, MA: MIT Press), pp. 469–490.