Donald Byrd
Advanced Music Notation Systems, Inc.
13 Detroit Ave.
Troy, New York 12180 USA
don@cogsci.indiana.edu

Music Notation Software and Intelligence

As the saying goes, a little knowledge is a dangerous thing. This applies to music notation software as much as to anything else. The articles on this topic in this and the previous two issues of *Computer Music Journal* make it clear that notation programs can no longer be thought of as the toys that we believed them to be only a few years ago. Now that score preparation programs are growing up—perhaps this is even evidence that they *are* growing up—they're suffering from one of the typical problems of adolescence; they act as if they know everything.

Why is this true, considering how much today's better programs know about rhythm, complex chords, instrumentation, and so on? The reason is that all this knowledge isn't much compared to what there is to know about common-practice Western music notation (CMN). This is a very complicated subject with many exceptions and subtleties. Lacking the space for an extended discussion, I'll have to rely on a few dramatic examples to make this point.

Shown in Figures 1–4 are examples of cases in which famous composers of the classic-romantic period flagrantly violated important rules of music notation and yet produced results that are easily readable (playable) by most musicians. These examples are taken from my Ph.D. dissertation (Byrd 1984), which includes many more examples and a detailed discussion. In Figure 1, Johann Sebastian Bach changed time signature in the middle of a measure (from the *Goldberg Variations*)! Figure 2 shows a measure with no less than four horizontal positions for notes that are all on the same downbeat (taken from Johannes Brahms's Intermezzo op. 117, no. 1). The notes in the dotted quarter chords occupy three different positions; the first eighth note on each staff, in yet a fourth position, is also on the downbeat. Finally, Figures 3 and 4 are two very different ways of having two clefs in effect on a staff at the same time. The first is bizarrely obvious (from Claude Debussy's *La Danse de Puck*). The other—in the fourth measure on the lower staff—is so subtle that one really has to think about the 3/8 meter here (obvious everywhere else in the example) to see that the bass and treble clefs are both in effect throughout the entire measure (from Maurice Ravel's *Scarbo* from *Gaspard de la Nuit*).

Why do these peculiar pieces of notation arise in the music of these highly respectable composers? The interesting thing is that there is really nothing very strange going on in any of these examples. In fact, it is easy to imagine someone playing through Figure 2 or 4 without even noticing anything unusual—and a listener to any of these examples would surely not notice anything unusual. Bach could have written the first example without a change of time signature at all, but it would have required tuplets and would probably have been harder to read. All of the other examples could have been written without any unusual notation simply by adding a third staff, but then the music would have required more paper (expensive for the publisher) and perhaps page turning (annoying for the performer). The point is that the supposed rules of CMN are not independent; they interact, and when the situation makes them interact strongly enough, something has to give way. It is tempting to assume that the rules of such an elaborate and successful system as CMN must be self-consistent. A problem with this idea is that so many of the "rules" are, necessarily, very nebulous. Every book on CMN is full of vague statements illustrated by examples that often fail to make the rule clear, but if you try to make every rule as precise as possible, what you get is certainly *not* self-consistent.

Software designers are well-intentioned people, and they tend to think they can best help their users by having their programs do things automatically. This is true if the sofware knows enough that it can do the right thing almost all of the time. SMUT, my first notation program, made many assumptions that

Computer Music Journal, 18:1, pp. 17–20, Spring 1994

Figure 1. Example from the Goldberg Variations *of J. S. Bach, demonstrating a change of time signature in mid-measure.*

Figure 2. Example from Johannes Brahms's Intermezzo *op. 117, no. 1, in which notes to be played on the same beat have four different horizontal positions.*

Figure 3. Example from La Danse de Puck *by Claude Debussy, showing two different clefs in effect in the same staff at the same time.*

Figure 4. Example from the Scarbo from Gaspard de la Nuit *by Maurice Ravel in which two different clefs are in effect in the same staff at the same time for an entire measure (the fourth in the lower staff).*

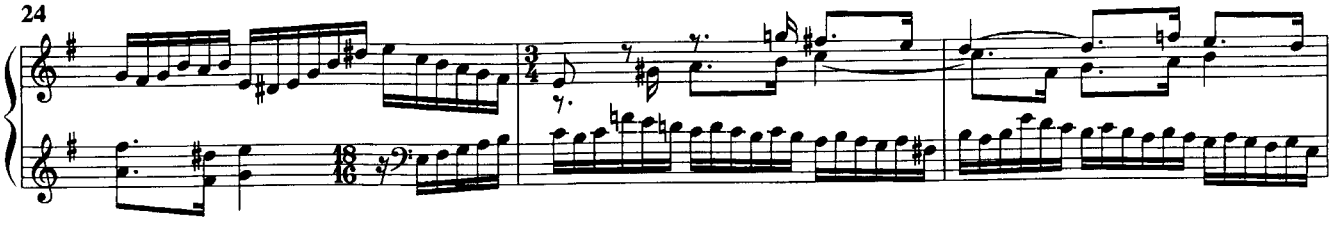

Figure 1.

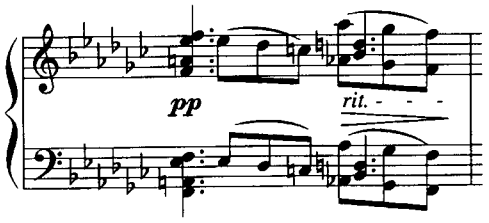

Figure 2.

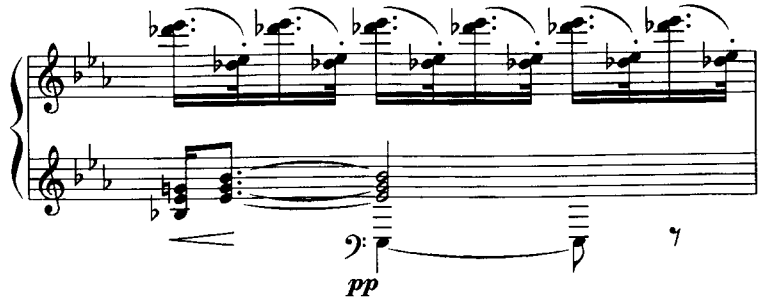

Figure 3.

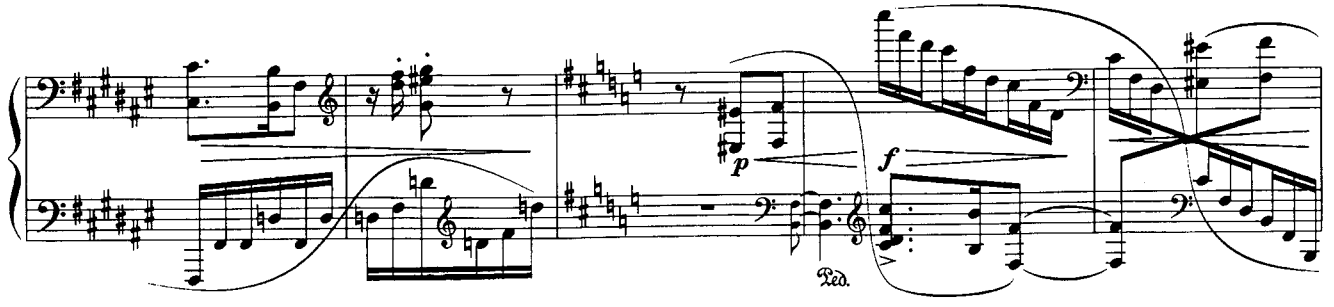

Figure 4.

are not always true (for example, that every voice in the score stays permanently on one staff). After working for years on it, I realized that it would have been more useful if it had not made so many assumptions, even though it would have been less automatic, and therefore less useful, when those assumptions were true. (Of course, best of all would have been a way to tell the program what it could assume; then users would not have to give up anything.)

Matters are made much worse by the fact that, these days, most notation programs attempt to convert CMN to performance—and vice versa—and therefore have to understand to some extent what the notation "means." If the software wasn't trying to play from notation, it could behave to the user's satisfaction with a lot less domain knowledge. A simple proof of the overwhelming difficulty of translating in only one direction—CMN to performance—is that instrumental and voice teachers in every music school spend a great deal of time teaching their students how to interpret CMN, and not all of this time is spent on subtle aspects that users wouldn't mind the computer overlooking. For example, in jazz and related styles as well as some baroque music, patterns of even eighth notes may correspond to very uneven played values—or they may not, depending on the tempo. The "interpreter" is supposed to know how they are to be played from his or her knowledge of the style.

Severo Ornstein, coauthor of the legendary (but never commercially available) program Mockingbird, has pointed out one of the worst offenses of many programs (1991):

> Right from the outset [most existing systems] assume the existence of a defined rhythmic structure (barlines, meter) as part of the staffing, onto which input is mapped (I would say forced).... [T]here is a ... serious problem that follows when music is represented in a rhythmically structured way. In working with a score on the screen, entering and removing material, all actions require the consequent material to fit somehow into the predetermined rhythmic structure. The consequences of actions that would tend to violate this structure must be forced into it in one way or another, and the resultant side effects may well be at variance with the user's intentions.... When the user puts a note down, he doesn't want it to be moved somewhere else just because the program thinks it understands where it fits into the structure. And he doesn't want it to cause anything else to move around either. He just wants the note to go where he put it. And when he deletes a note, that's all he wants to happen.... He may well plan to use the space thus freed up for something else. It's disturbing to have things moved into space you just tried to clear, things that don't belong there and which then lose their alignment with the things they do belong with.

Conclusion

In my dissertation, I compared music notation to Chinese writing and to mathematical notation and argued that CMN is vastly more complex than Chinese. Chinese has a very large character set, but almost all you do with the characters is arrange them in rows and columns and start new lines and pages as needed, much as in any other language. In music, layout is a major graphics problem, and there is no fixed character set, no matter how large—consider beams and especially slurs (and what about accidentals, articulation marks, and augmentation dots: are they independent characters, or are they parts of some single character along with the note to which they belong?). Mathematics is a more worthy opponent, but music almost certainly is still more complex.

I concluded, "Much music exists whose correct formatting requires considerable intelligence (well beyond the state of the art of artificial intelligence), and some music exists whose correct formatting probably requires full human intelligence." In my dissertation, I argued "the nonfeasibility of fully-automatic high-quality music notation (FAHQMN)"; this position was inspired by Bar-Hillel's famous (1960) paper on "the nonfeasibility of fully-automatic high-quality translation" (referring to natural-lan-

guage translation). Bar-Hillel wrote in the face of widespread over-optimism about the tractability of the problem he discussed. The situation is somewhat similar with music notation, though an approximation that is good enough to be useful is probably easier than the equivalent for translating natural language. I see no reason to change these statements today. Transcribing and playing from notation, as most programs on the market today do, is much more demanding than just correct formatting. CMN is too difficult to handle automatically—at least until computers get a great deal smarter. In the meantime, any music notation program that thinks it can tell what its users really want is going to do them a lot of favors they would have been better off without. I'm not saying we need programs that know less, just ones that know how much they don't know.

Acknowledgments

My thinking on user interfaces for music notation systems has been influenced tremendously by John Maxwell and Severo Ornstein, the creators of Mockingbird. I've also learned a lot from Pat Billingsley, Bill Buxton, Ric Ford, John Gibson, Jim Hettmer, Doug Hofstadter, Jef Raskin, the designers of the Macintosh user interface (including Raskin), and others. I'd also like to thank Pat Billingsley and Steve Larson for their comments on this article.

This article is an extended revision of a text that appeared as an introduction to the chapter on music notation tools in Christopher Yavelow's exhaustive *Macworld Music and Sound Bible*, published 1992 by IDG Books

References

Bar-Hillel, Y. 1960. "A Demonstration of the Nonfeasibility of Fully-Automatic High-Quality Translation" Appendix 3 to "Present Status of Automatic Translation of Languages" in F. L. Alt, Ed. 1960. *Advances in Computers* Vol. 1. London: Academic Press pp. 158–163.

Byrd, D. 1984. "Music Notation by Computer." Ph. D. diss., Indiana University Computer Science Department.

Ornstein, S. 1991. Private communication with the author.