Creating Printed Music Automatically

Gary M. Rader
Mesa State College

The quality of printed music produced by software programs depends on the user's knowledge of copyist rules. This constraint-based technique automates uncomplicated, common notational tasks.

Some of today's music-score editing programs are capable of producing publication-quality scores. Some, such as Score (San Andreas Press), Finale (Coda), and Encore (Passport Designs), contain many powerful automation features in addition to notation capabilities. However, to produce scores as good as those music engravers and copyists create, the users must understand the positioning of many notational elements. For instance, should a *slur* appear above or below the affected notes? Should a *tuplet* appear between the slur and the notes or outside it?

Scores produced by users who do not know the rules not only look wrong, they are difficult for musicians to read. Even if performers are not consciously aware that something is wrong, their subconscious reactions, conditioned through a lifetime of practice, will not be the same. In the past, a performer might have recognized and dismissed nonstandard notation as just amateurish. However, modern music features so many new and radically different notational systems that errors could become serious obstacles to the correct interpretation of a composition.[1]

Most users of notational software have little or no music copyist training, and musicianship does not guarantee notational skills. Even professional composers fail at many notational tasks if they have not had explicit copyist training. The ability to notate music has very little in common with the ability to compose or play it.

Unfortunately, we do not have the rules for notating music in a prescriptive form, that is, one that a computer can use. Music copyists take years to learn them. For the most part, books present these rules in descriptive form using examples.[1-3] Individual rules by themselves present little problem to notational software. It is the interaction between the rules that is so difficult to capture.

This article presents a partial solution to this problem: How can users with little or no copyist training create publication-quality printed music? The solution, a program called MusicEase, applies constraints to notational information the user enters, automatically arranging graphical elements "correctly." This can also speed up the notation process, since the copyist doesn't have to make as many decisions or enter as much data.

MusicEase incorporates a constraint-based technique to calculate the complex interaction of notational elements. Basically, it is a WYSIWYG editor with a menu system for novice users and keyboard shortcuts for advanced users. It contains most of the features of modern word processors, such as zooming and page preview. It also imports and exports standard Musical Instrument Digital Interface (MIDI) files. I wrote MusicEase using muLisp. Al Rich and David Stoutmeyer of the Soft Warehouse in Honolulu created muLisp, an MS-DOS version of Lisp. muLisp incorporates many of the ideas of Common Lisp and other modern Lisp systems,

0018-9162/96/$5.00 © 1996 IEEE

but remains compact enough to run under MS-DOS's limited memory space.

NOTATION AT THE SYSTEM LEVEL

The excerpt from Ludwig van Beethoven's "Harp" string quartet in Figure 1 illustrates some of the interaction between notational elements. The fragment, generated in MusicEase, shows three bracketed *systems*, or groups of one or more staves. Each system in this example contains four staves. All systems in Figure 1 have the same number of *measures*—three—although different systems can have different numbers of measures. The "Music terms" sidebar defines some of the vocabulary for easy reference.

A *staff* consists of five horizontal lines. Each staff begins with a *clef sign*, followed by a key signature that specifies the context for the pitches of the staff's notes. Each staff generally indicates the part for a different instrument: first violin, second violin, viola, and so on. In the case of a keyboard instrument such as a piano, two separate staffs might represent the parts for each hand.

A basic musical *note* consists of pitch and duration. A written note uses an oval *notehead*. Its vertical position on the

Music terms

Accent mark—Indicates to play the note with emphasis. Examples include staccato (brief) and tenuto (long). (See *articulation sign*.)

Accidental sign—Indicates to raise or lower the pitch specified by the note following the sign.

Articulation sign—Indicates how a performer should blend successive notes together. (See *accent mark*.)

Beam—A thick horizontal line that substitutes for individual flags in a group of flagged notes.

Casting off—The process of breaking the *systems* of a piece so that they fit within the left and right margins (analogous to line-breaking in text).

Chord—Three or more *notes* played at the same time, indicated by a number of noteheads on one stem.

Clef sign—Appears at the beginning of each *staff* in a *score* and denotes which *notes* (pitches) the various lines and spaces of the *staff* represent. The treble clef (looks roughly like an ampersand or a script "S") and bass clef (looks like a backwards "C" with two dots) are very common and appear in the figures of this article.

Dynamics—Symbols to indicate changes in volume, such as "***p***" for piano, "*cresc.*" for crescendo, or a wedge (for crescendo or decrescendo, depending on the direction in which it points).

Justification—The act of stretching or shrinking a *staff* or *system* horizontally so that it extends from the left margin to the right margin. Some symbols, like noteheads, are not physically altered, only the spaces around them. Other symbols, like beams and slurs, are actually stretched or shrunk. (Analogous to justification in text.)

Ledger lines—Short horizontal lines that lie beyond the five lines of a *staff*. These represent an extension of the staff and facilitate the reading of *notes* positioned outside the normal staff lines.

Lyrics—Syllables or phonemes associated with *notes*. The performer sings these at the pitch and for the duration that the notes indicate.

Measure—A segment of a *staff* that specifies a unit of time comprised of a fixed number of like note durations (all whole or half or quarter and so on), depending on the *meter*, and bracketed by two vertical lines. Each measure in meter 4/4, for example, will contain four beats, one for each quarter note.

Meter—The time signature of a *score* that indicates the number of beats in a *measure* and the *notes* that receive an accent within each measure. Common meters are 4/4 (fox-trot), 6/8 (polka), and 3/4 (waltz).

Note—Used to specify the duration and pitch of each sound in a *score*. The durational terms "whole note," "half note," "quarter note," and so on, in conjunction with the *meter*, indicate how many beats these notes represent. The position of each note on the *staff* indicates the actual pitch to be produced. This term is also used to describe the sounds produced by an instrument when playing the sounds indicated by written notes.

Overlay—A nonstandard term I use to describe a genre of symbols that extend over a group of *notes*, such as the wavy line of a trill or the dashed line of an octave transposition.

Phrase mark—A curved line placed over or under a segment of musical structure that is unified by rhythms, melodies, and harmonies, and that comes to a closure. The performer should articulate the notes in the segment smoothly as a unit.

Rest—Indicates that the performer should not play or sing for the duration of the notational device.

Score—A written piece of music that shows the *notes* to be played or sung by each instrument or voice. Sometimes the terms "song" and "piece" are used as synonyms for "score."

Slur—A curved line placed over or under a group of *notes* of differing pitches to indicate that the notes should be played without articulating them separately.

Staff—A set of five equally spaced horizontal lines. *Notes* written upon these lines and spaces indicate pitch. The plural is staves.

System—A connected group of two or more *staves* whose *notes* are played simultaneously, as in orchestral, choral, and piano music.

Tie—A notational convention that extends the duration of a *note* by connecting an arcing line from the note to the following note. The second note must have the same pitch as the first. The first note is sounded for the combined duration of both notes, and the second note is ignored.

Triplet—A common form of *tuplet* that consists of three *notes* played in the time that two notes usually take.

Tuplet—A group of *notes* played in the time it normally takes to play a proper subset of the notes. A 7-tuplet of eighth notes would be played in the time normally used for six eighth notes.

Voice—The *notes* each different instrument should play or each singer should sing. Often, each *staff* contains the part for one voice.

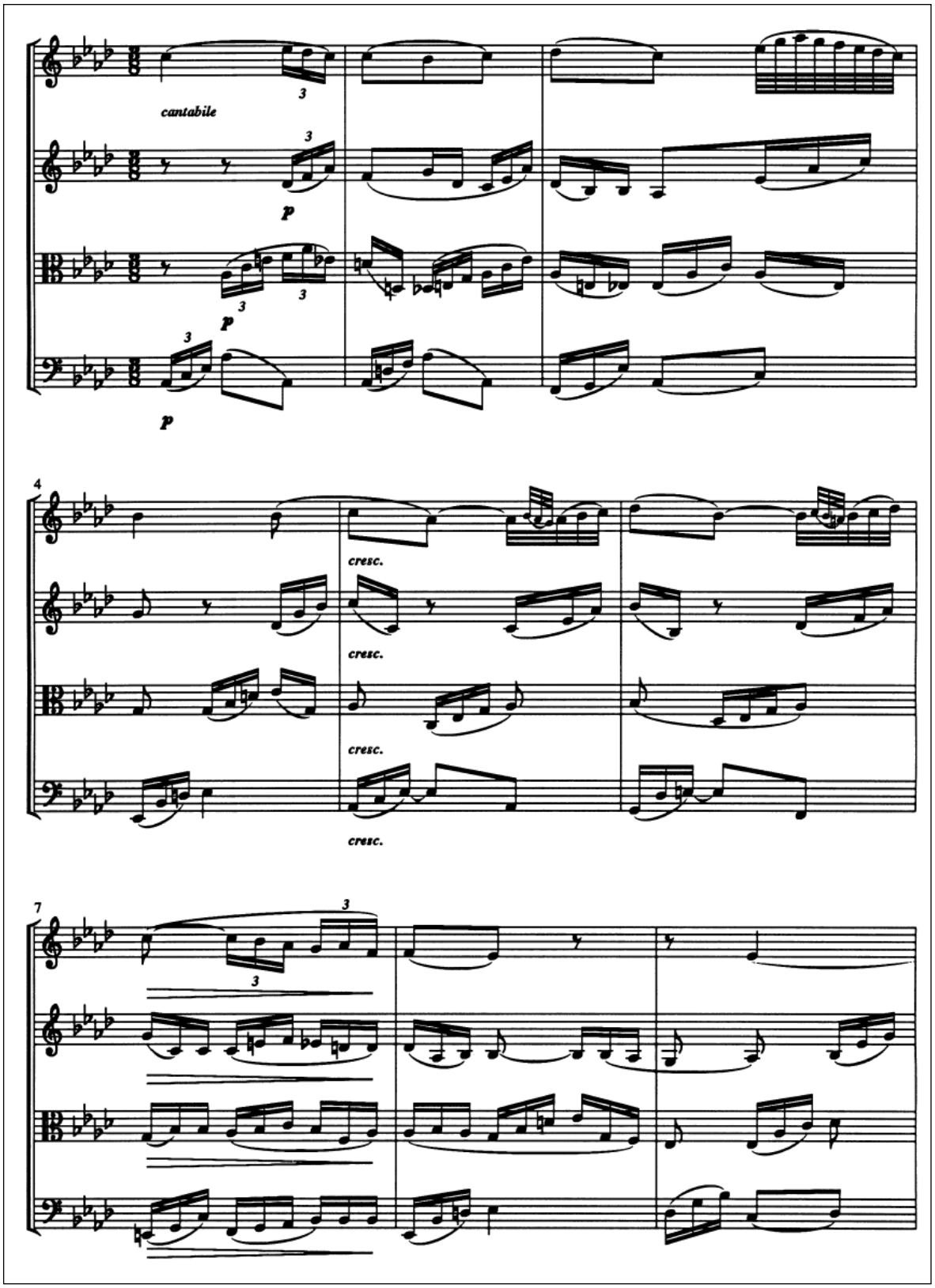

Figure 1. Excerpt from Beethoven's "Harp" string quartet.

staff and an optional *accidental sign* identify the pitch it represents. When the pitch represented by a notehead lies outside the staff's range, it requires extra *leger lines*. MusicEase creates the leger lines automatically when the user enters a note beyond the staff's range. Depending on its duration, the notehead can be hollow or solid. It can have a vertical *stem*, perhaps ending with a curved *flag*. If one or two dots follow the note, that indicates an increase in duration. *Rests*

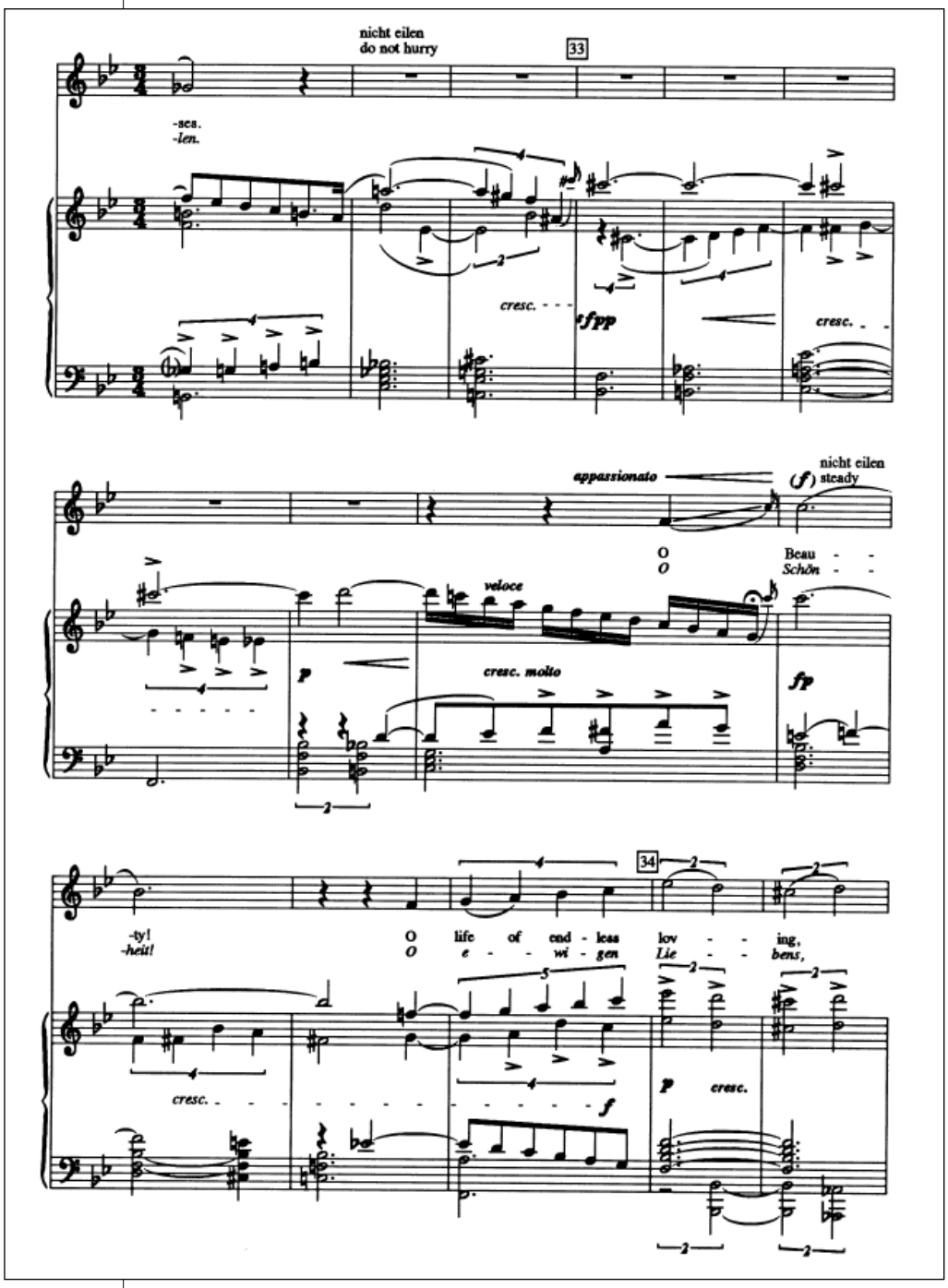

Figure 2. Fragment from Mahler's "Das Lied von der Erde."

occupy places where no note is sounding. They come in different shapes depending on their durations. We express basic durations as fractions of a whole note, which is represented by a hollow oval with no stem. Figure 1 contains notes ranging from one quarter (stem but no flag or dot) to one sixty-fourth (four beams) in duration, but only eighth note rests. Smaller grace notes should be played relatively fast. By notational convention, grace notes are considered to have no duration.

As Figure 2, a complex fragment from Gustav Mahler's "Das Lied von der Erde," shows, a number of notes can occupy the same horizontal position on a staff, while having different vertical positions. This is called a *chord* instead of a note. All notes in a chord are sounded simultaneously.

Lyrics are words associated with notes. Usually a score hyphenates them and centers the syllables or phonemes under the notes with which they are associated. The performer will sing the lyrics at the pitch and duration specified by the note.

When we want to maintain a note across metric boundaries, we can follow it with another of the same pitch (although perhaps a different duration), connecting the two by a curved line called a *tie*. Vertical bars usually divide each staff into measures. The sum of the durations of the notes and rests in each measure is the same for all measures with the same time signature. Abbreviations such as "***p***" and "*cresc.*" (short for "piano" and "crescendo") and horizontal *wedges* (such as an expanding wedge indicating a crescendo) specify *dynamics*. These various elements interact in complex ways, which MusicEase handles by applying a set of constraints.

Elements of music notation

Besides the basics of staves, notes, and rests, a score can indicate how a performer should play the notes. This section generally discusses staves that contain a single voice. Staves containing several voices (double-stemmed notation) require comparable treatment. Upper-voice notation appears higher or above the staff, while lower-voice notation usually appears lower or below the staff. For example, all staves in Figure 1 are single-voice notation, while many of the staves in Figure 2 contain double-stemmed notation. Here I use the word "note" to mean either a note, a chord, or a rest.

Flagged notes are often joined by one or more horizontal *beams* (which replace the flags) to convey to the performer a better feel of the rhythm. Beams combine shorter notes into groups for easier reading. The groupings depend on the

durations of the notes, their placement within the measure, and the *meter*. A note has a downward stem if its notehead lies above the middle staff line. If one note in a group has an upward stem, the beam runs above the grouping. Otherwise it appears below the grouping. Beams are half as wide as the space from one staff line to the next. Generally the beam slant should follow the general trend of the grouped noteheads, but at a shallower angle. However, if the noteheads are much closer together than normal because of tight spacing, the beam is horizontal. Beam slant should never be steeper than about 30 degrees. Music copyists learn the exact rules for beam slant and stem length from a large number of examples.[4] The beam ends must either sit on, straddle, or hang from the staff lines. That is, beam ends must touch a staff line—an end cannot hang in space between two lines. Double beams follow similar rules and have a quarter of a staff space between them. Triple and quadruple beams also follow similar rules, but on occasion we might reduce the beam thickness slightly. Most beamed groups need to extend or shorten the stem lengths of some notes to line up with the beam. However, all stems must meet a minimum length. The stems in beamed groups always extend to or cross the middle staff line. Figure 3 shows some examples of beamed groups of notes.

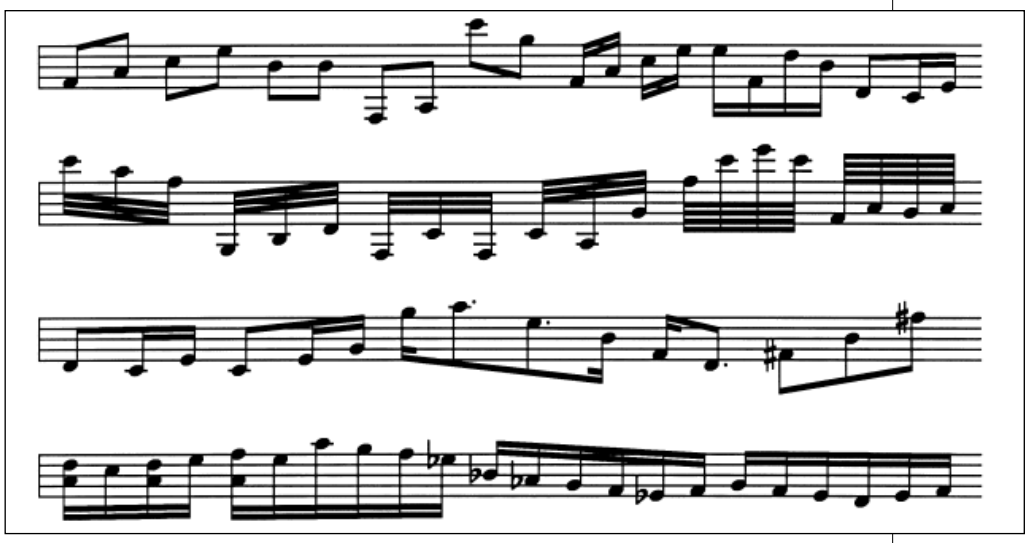

Figure 3. Examples of beams.

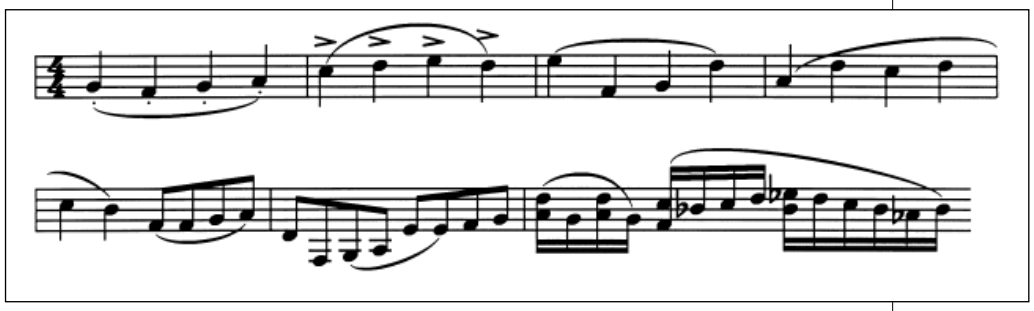

Figure 4. Examples of slurs.

Slurs generally extend over or below a group of notes to indicate phrasing, bowing, or breathing. If at least one note in the slur has a downward stem, the slur appears above the notes. Otherwise it appears under the notes. The same rule applies to ties. As with beams, the general slant of the slur should follow that of the noteheads as much as possible. Unlike beams, slurs (and ties) can continue from the end of one staff to the beginning of the corresponding staff in the following system. Such cases place the slur or tie above or below the group according to the positions of all the notes.

Generally, *articulation signs* should lie between the slur and the affected notehead, as in the first measure in Figure 4. A period indicates *staccato*, a shortened note; a dash indicates *tenuto*, a lengthened note. In the figure, the characters re-sembling greater-than signs indicate a percussive ac-cent. When the starting or ending note of a slur has a percussive accent, the slur should generally appear between the note and the *accent mark*, as in the second measure in Figure 4.

Triplets indicate that three notes should be played in the time in which two notes are usually played. A small numeral "3" and an optional bracket above or below the notes marks a triplet. This concept, generically called a *tuplet*, extends to any number of notes. Tuplet brackets extend over or under a group of notes according to the rules for slurs. If the tuplet notes are fully beamed—the beam start and end correspond with those of the tuplet—we can omit the tuplet bracket. The slant of any tuplet brackets should follow the general slant of the noteheads, just as with slurs and beams. If all stems lead down, the bracket runs below the staff. If all stems point up, the bracket lies above the staff. Authorities disagree somewhat on where to place the bracket when stems lie in both directions. Much depends on the amount of articulation and phrasing: You should position the bracket where it interferes the least with these. For instance, vocal music always places brackets above the staff, even if the notes are fully beamed and the stems point down, to make room for lyrics.

The numeral that indicates the number of notes in the tuplet sits within the bracket, or outside the beam if there is no bracket. If the tuplet has only two short notes, the numeral should be centered horizontally between the notes. Again, some disagree on the numeral's horizontal placement in other situations. One view is that it should lie at the tuplet's durational center. Another is that fully beamed groups should center the tuplet numeral on the beam and that musical context dictates its placement oth-

erwise. A third suggests compromising between the two, depending on context.

A slur that affects any notes in the tuplet and lies on the same side as the tuplet bracket should go between the tuplet and the notes. For example, measure three of system one, staff two, in Figure 4 shows a tuplet bracket marked "4," then a tie, then a slur. Similarly, the tuplet brackets go outside of any articulation marks. Sometimes many tuplets consisting of the same durations, such as the sixteenth-note triplets in Figure 1, occur in a piece. When there's no chance of confusion, you can omit the tuplet numerals and brackets from most tuplets to improve readability.

Applying constraints

Many constraints depend on the results of applying other constraints. MusicEase positions items according to their order in this list:

1. Noteheads
2. Beams and stem directions and lengths
3. Articulation symbols that, if a slur is present, generally lie between the slur and their associated noteheads (such as the tenuto in Figure 5h)
4. Slurs
5. Articulation symbols that, if a slur is present, generally lie on the side of the slur away from their associated noteheads (such as the second "greater than" percussive accent in Figure 5h)
6. Tuplets
7. Phrase marks
8. Crescendo wedges
9. Overlays

This list order reflects the way the items affect each other.

Postition dependencies. The positions of slurs depend on the positions of simultaneously occurring beams. The positions of tuplets depend on those of slurs. However, the reverse is not true: The position of a beam never depends on the position of a slur. If a staff contains beams, slurs, and tuplets, MusicEase calculates the positions of beams first, then slurs, then tuplets, regardless of the order in which the user entered these items. If the user specifies beams and then tuplets, they will appear correctly positioned. If the user then enters slurs, the tuplets will shift automatically to make room for them.

Some articulation symbols lie between the slur and notehead except when they occur at one end of the slur. For example, in Figure 5h, the second "greater than" accent lies outside the slur (end), while the third lies within the slur. Depending on the context, such symbols fall into either category 3 or 5 in the list above. The main factor differentiating symbols in 3 and 5 is that smaller symbols usually fall into category 3, while larger symbols often fall into category 5.

Before redisplaying a staff, the program updates information it maintains about the staff positions of notational elements. For a notehead, it saves the x and y coordinates of a crude outline. Anything positioned with respect to the notehead follows these points. For a stem, the program first determines the correct side, then computes an initial length. It then determines the stem position. If the stem goes up, the program aligns the bottom of the stem with the point on the right side of the notehead. If it goes down, the point on the left side marks the position for the top of the stem. MusicEase uses the coordinates of the stem's end points to properly display beams and other notational elements whose positions depend on the stem end. If the stem later needs to be lengthened or shortened to connect to a beam, the program changes the y coordinate accordingly. In the case of a chord, MusicEase saves the coordinates for all noteheads and uses the appropriate coordinates when attaching a stem. It handles other elements similarly.

Generally, the most complex elements to position are slurs and *phrase marks*, since they must curve around intervening elements. MusicEase uses Bezier splines to generate these curves and draw ties.[5] To generate a slur or phrase mark, the program makes an initial guess at the shape. This guess considers the starting and ending notes and whether the curve goes above or below the notes. If the curve does not pass around all intervening elements, the program extends the two interior points specifying the Bezier spline. It then checks the new curve to see if it passes around all intervening elements. This process continues until either it succeeds or the curvature becomes too round, forcing the program to alter one or both end points. The technique for splitting a curve over several staves is similar, except that the default points are staff ends, not notes. For output, the program generates two curves close together with roughly the same end points. It then fills in the interior to simulate a pen stroke. As an example, MusicEase generated the slurs in Figures 1 and 4 automatically.

Constraint application. The following example is designed to give an idea of how MusicEase applies constraints to a number of symbols that can occur simultaneously. While it does not make much sense musically, it shows how adding notational elements automatically repositions other elements.

Figure 5a shows a fragment of a staff containing two voices. In Figure 5b, the user specifies a triplet. This tells the program to automatically change the beaming and place the number "3" above the beam. The triplet does not receive a bracket because its scope corresponds to the scope of the beam. Next the user adds an octave transposition, as Figure 5c shows. The transposition goes above both the beam and the triplet numeral because of the interacting constraints. Figure 5d includes a phrase mark that extends from the first to the last note of the top voice. The octave transposition moves up to make room for the phrase mark. In Figure 5e the user inserts a slur that extends from the third note to the last note. The triplet numeral, phrase mark, and octave transposition all move up to make room for the slur. Figure 5f shows the addition of a tenuto to the third note. The left end of the slur automatically moves up. The user then appends a fermata to the last note (Figure 5g). This necessitates shifting the right end of the phrase mark up. Finally, the user adds accent marks to the first, third, and fifth notes, as Figure 5h shows. The program moves the center of the slur up, along with the triplet numeral. The center and left end of the phrase mark also automatically shift upward.

The resulting score fragment would probably be too complex for a naive user to notate correctly. However, here the program does it all automatically using constraints. The basic positions of the notational elements are correct from a music copyist's viewpoint.

NOTATION AT THE SCORE LEVEL

Besides measure-level notation, copyists face the problems of *justification* and *casting off*, analogous to justification and line breaking in typesetting. Systems should generally break to align similar groups of notes horizontally throughout the entire score. And unlike text, the last system must extend to the right margin and be just as dense as previous systems. The last page should often also be completely full. It cannot, for instance, contain only one or two staves. This ignores special concerns such as ending a page in a place where the performer can turn it without falling behind. In addition, different categories of instruments (vocal, harp, woodwind, string, and so forth) have their own notational idioms.

Notes in a system that sound at the same time must line up vertically. The horizontal spacing of notes roughly corresponds to their relative occurrence in time. This helps performers visualize the timing of the start and end of notes. All systems are force-justified. This generally requires increasing or decreasing the spacings of all notes so that staves extend from the left margin to the right margin. Any slurs, beams, wedges, and other markings must be stretched or shrunk to fit.

MusicEase applies constraints as often as feasible. As the user enters notational elements, the program automatically changes the positions and shapes of other elements in the vicinity. For example, a slur might expand to fit around a newly entered symbol. When the user adds notes in a bottom voice, the program flips downward-arcing ties in the top voice so that they point up. However, in some cases the immediate application of constraints is not feasible. For instance, MusicEase performs justification only when the user asks. Otherwise, notes on the display would constantly jump around as the user entered notes on the same staff or on other staves in the same system. Casting off occurs only upon user invocation, since it generally does not make sense to casting off until the user finishes entering the complete piece. Similarly, the program only vertically justifies systems when the user invokes printing or print preview.

Casting off

Casting off is the process of breaking staves or systems so that all measures have approximately the same density after justification. For instance, two measures that contain exactly the same elements should have the same horizontal spacing after justification. Casting off also encompasses setting page breaks to minimize the disturbance caused by turning pages.

Copyists approach this problem by trial and error. They first decide which size of elements to use. Then they create a first draft of the layout. They then adjust the result, tightening or loosening spacing for the whole piece or sections of it, until the score meets the requirements. A scoring program that alters system breaks must be able to break slurs, ties, phrase marks, and wedges across system boundaries.

MusicEase lets users specify the maximum number of measures in a system, applied to the whole piece or a block. Then it compares the minimum acceptable width of a system after justification with the width between the margins. (This computation includes any indentation, such as that used for the first system.) If the minimum system width is greater than the allowable space, MusicEase automatically reduces the number of measures until the system satisfies this constraint.

MusicEase also incorporates a more general version of casting off based on the procedure developed by Wael A. Hegazy and John S. Gourlay.[6] This provides results comparable to those a copyist produces, except that it neither pays attention to where pages break nor fills the last page. However, it does fill the last system, and it does maintain a uniform density throughout.

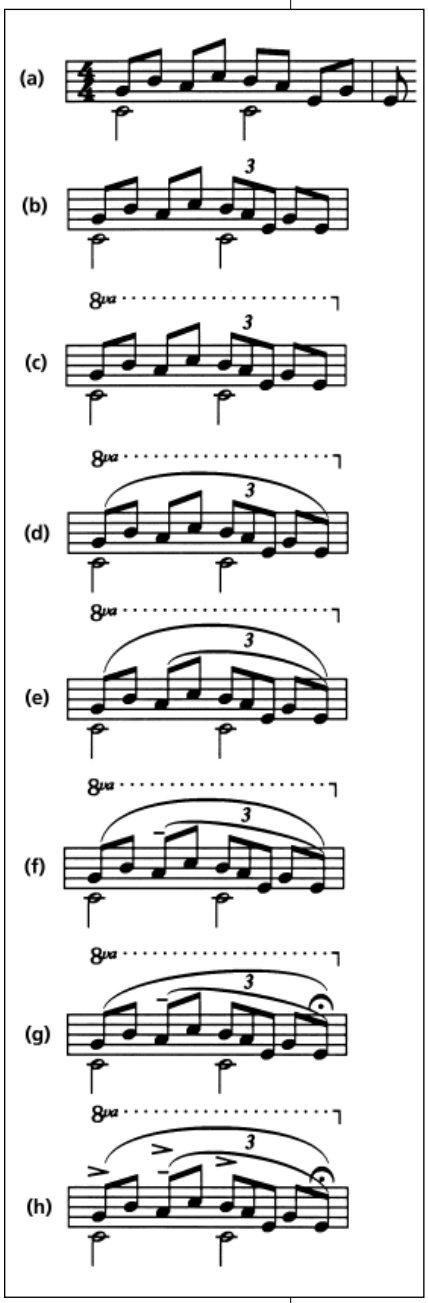

Figure 5. Successive applications of constraints.

Justification

Justification stretches or shrinks the horizontal spacing of system elements to extend from the left margin to the right margin. Basically, spacing is logarithmically proportional to the durations of the notes. However, certain elements can force a note to require more space. Accidentals could require increasing the space between a note and the note that precedes it. A lyric syllable could require that the space between its note and the notes both preceding and following it be increased. Ties, for instance, have a minimum length, which could force the space between two notes to increase.

The spacing of one note affects the spacings of the other notes that sound at the same time. This can grow somewhat complicated when tuplets are involved. For example, if a seven-tuplet appears against six notes, each note in the tuplet should get 6/7 of the space an ordinary note gets. With multiple tuplets and other factors occurring simultaneously, the notation can become quite complex.[7]

MusicEase performs justification in two passes. The first

calculates the ideal spacing for each note or chord and its elements, plus meter changes, clef changes, and so on, for the entire system. The second calculates how much to shrink or stretch each note's space to create a fully justified system. If the user didn't run the casting off routine to determine system breaks, a maximally shrunken system might still extend beyond the right margin. (Users can check for this easily using the print-preview feature.) Also, successive notes can get only so close to each other and remain legible.

The justification algorithm is a slight extension of that presented by Gourlay.[8] Gourlay's algorithm does not consider a number of minor things like minimum tie lengths or when justifying a system requires exceeding the maximum stretch for notes. I only discovered these deficiencies after applying the algorithm to enough different types of pieces.

Since both casting off and justification routines consider the size of elements, MusicEase users can easily experiment with the results after scaling. For example, say a song's lead sheet fills a page, with one system falling on a second page. The user can scale the elements down slightly and reapply casting off until the result fits nicely onto a single page. A similar approach works for larger pieces. Users can manually specify page breaks or scale the piece larger or smaller to fill the last page.

MUSICEASE HAS BEEN IN USE at the University of Colorado at Boulder and other places for several years now. According to user feedback, the word-processing paradigm and constraint-based approach work reasonably well for many routine, uncomplicated, notational tasks. Many simpler pieces require little or no overriding of the automatic formatting. It relieves naive users from handling most notational details in simpler pieces, letting them quickly produce publication-quality scores without copyist knowledge. You can download a PK-Zipped shareware version (me423.zip) from several forums on CompuServe or by anonymous ftp from ftp.secret.com/ad-ftp/ (an OS/2 server). The current version of MusicEase runs under MS-DOS, but a Windows version should be available this summer.

However, more complex notational tasks are beyond MusicEase's current capabilities. For example, MusicEase's constraints handled the first system in Figure 1 satisfactorily. But the second system required some user intervention in the areas where grace notes occurred. I also created Figure 2 using MusicEase, but the automatic positioning of some elements was unsatisfactory. I had to correct them manually. Further research will show how more thorough specifications of existing constraints would improve the results in such situations. However, adequately addressing all forms and periods of music notation will require many more constraints. ■

References

1. K. Stone, Music Notation in the Twentieth Century, W.W. Norton., New York, 1980.
2. G. Heussenstamm, The Norton Manual of Music Notation, W.W. Norton, New York, 1987.
3. G. Read, Music Notation: A Manual of Modern Practice, Taplinger Publishing, New York, 1979.
4. T. Ross, The Art of Music Engraving and Processing, Hansen Books, San Francisco, 1970.
5. F. Sola, "Computer Design of Musical Slurs, Ties and Phrase Marks," Tech. Research Report OSU-CISRC-10/87-TR32, Ohio State Univ., Columbus, Ohio, 1987.
6. W. Hegazy and J. Gourlay, "Optimal Line Breaking in Music," Tech. Research Report OSU-CISRC-8/87-TR33, Ohio State Univ., Columbus, Ohio, 1987.
7. D. Blostein and L. Haken, "Justification of Printed Music," Comm. ACM, March 1991, pp. 88-99.
8. J. Gourlay, "Spacing a Line of Music," Tech. Research Report OSU-CISRC-10/87-TR35, Ohio State Univ., Columbus, Ohio, 1987.

Gary M. Rader *is an associate professor of computer science at Mesa State College, Grand Junction, Colorado. His research interests include artificial intelligence, user interfaces, software design, and computer music. He has worked at Bell Laboratories, been involved with several entrepreneurial companies, and has taught at the University of Ife, Nigeria; Eastern Washington University; the American University in Cairo; and Bradley University. Rader received a BA in mathematics, an MA and a PhD in computer and information science from the University of Pennsylvania, and an MBS from the University of Phoenix.*

Address questions about this article to Rader at the Department of Computer Science, Mathematics and Statistics, Mesa State College, PO Box 2647, Grand Junction, CO 81502; grader@mesa5.mesa.colorado.edu.