Virtual Music

Computer Synthesis of Musical Style

David Cope

With commentary by Douglas Hofstadter
And with perspectives and analysis by
Eleanor Selfridge-Field, Bernard Greenberg,
Steve Larson, Jonathan Berger, and Daniel Dennett

1 Virtual Music

Virtual music represents a broad category of machine-created composition which attempts to replicate the style but not the actual notes of existing music (Cope 1993). As will be seen, virtual music has existed in one form or another for centuries. With the advent of computers. however, the potential for virtual music has multiplied exponentially. In this chapter, I provide a brief background of virtual music and then ask you to participate in three listening tests which will challenge your ability to recognize human-composed vs. computer-composed music and to recognize actual Bach and Chopin vs. computer-composed music in their styles.

Early Examples

The figured bass, popular during the Baroque period of music history (1600–1750), demonstrates how composers and performers use combinations of notated music, period style constraints, and performer choice to produce a diversity of results and yet adhere to a composer's style. As in other examples of virtual music, each performance differs, yet each retains its stylistic integrity.

Figured basses constitute the notation for most Baroque basso continuos, the combination of a keyboard instrument (clavier or organ) and a reinforcing sustaining instrument (bass gamba, violoncello, or bassoon). Typically the keyboardist uses a notated bass line, or a bass-line and treble-line depending on the ensemble requirements, above which they freely but stylistically improvise.

Figure 1.1 shows a very simple figured bass in C major. The arabic numerals below certain notes indicate inversions of chords. Performers assume root position or 5/3 intervals above the bass-note unless otherwise instructed. The bass gambist or cellist plays the line as written. The keyboardist, however, must complete the implied chords in the proper key in a manner consistent with the style, yet original in spirit. In essence, the figured bass represents an algorithm or recipe, the realization of which depends upon the application of performance practice and performer style improvisation.

Figure 1.2 provides a very simple realization of the figured bass of figure 1.1. The chords here consist of triads (three-note chords built in thirds) or seventh chords (four-note chords built in thirds) with some notes doubled in octaves. The Baroque period constraints governing which notes should be doubled, as well as how notes should move, one to another, are quite strict and too numerous to present here. The important thing, at least for our purposes, is to understand that the music in figure 1.2 represents only one of many possible realizations of the figured bass of figure 1.1.

Figure 1.3 shows another possible correct realization of the figured bass shown in figure 1.1. Again, the music here consists of triads and seventh chords with some

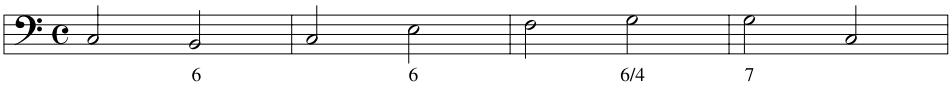


Figure 1.1
A simple figured bass in C major.

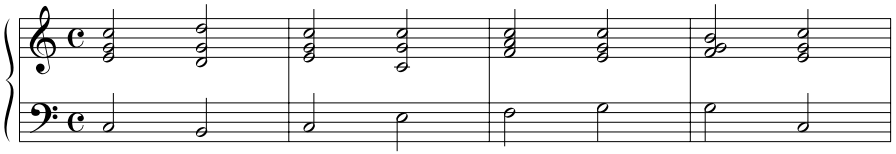


Figure 1.2
One possible realization of the figured bass in figure 1.1.

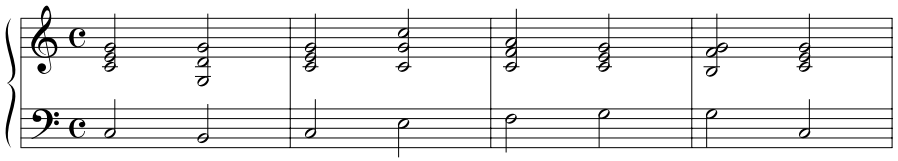


Figure 1.3
Another correct realization of the figured bass shown in figure 1.1.

notes doubled. Comparing figure 1.3 with figure 1.2 demonstrates both their similarity (same note names in each chord) and differences (notes in different registers). In essence, then, we have two different examples of music, in similar block chord style, derived from the same core figured bass.

Figure 1.4 presents a more Baroque-style realization of the figured bass in figure 1.1. In fact, the melody shown in this example might typically be one of the provided elements. While this example tends to resemble figure 1.2 in chord spacing it nonetheless represents a third and distinctly different realization of the figured bass in figure 1.1. In all of these cases, the music has adhered to the constraints of the period using a combination of given music and performer choice, as well as a recombination of right notes and motives.

The *Musikalisches Würfelspiel*

One of the first formal types of algorithms in music history, and another good example of virtual music, is the eighteenth-century *Musikalisches Würfelspiel*, or musical dice game. The idea behind this musically sophisticated game involved

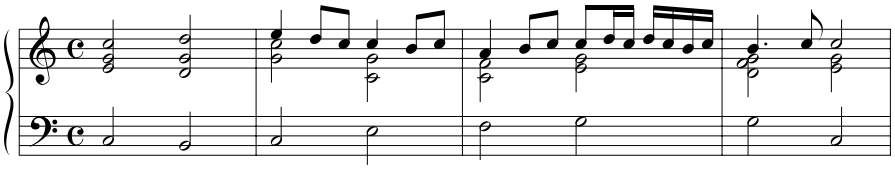


Figure 1.4
A more typical realization of the figured bass presented in figure 1.1.

composing a series of measures of music that could be recombined in many different ways and still be stylistically viable—virtual music. Following this process, even a very simple piece becomes a source of innumerable new works. A typical *Würfelspiel* of sixteen measures, for example, yields 11^{16}, or roughly forty-six quadrillion works, with each work, although varying in aesthetic quality, being stylistically correct (Cope 1996). Composers of *Musikalische Würfelspiele* included Johann Philipp Kirnberger, C. P. E. Bach, Franz Josef Haydn, Wolfgang Amadeus Mozart, Maximilian Stadler, Antonio Callegari, and Pasquale Ricci, among others (see Cope 1996).

Figure 1.5 provides an example of a matrix from a typical *Musikalisches Würfelspiel*, this one attributed to Franz Josef Haydn. The numbers down the left side of the matrix in figure 1.5 represent the eleven possible results of the toss of two dice (2–12). Each number in the matrix links to a previously composed measure of music. Each vertical column of the matrix indicates successive measure choices (A–H here representing an eight-measure phrase). To get a first measure of music, one tosses the dice, locates the resulting number on the left of the matrix, and then looks up the corresponding measure in vertical column A in an associated list of measures of music (not shown here due to space limitations). Subsequent tosses for columns B through H complete an initial phrase, with further phrases produced in the same way using different matrices and musical correlates. A resulting minuet appears in figure 1.6.

Composers of *Musikalische Würfelspiele* created the various measures in such a way that any of the measures in one vertical column would successfully connect with any of the measures in the column to their immediate right. This becomes fairly clear when the actual music for each measure is aligned as in the matrix. However, the music of a *Musikalisches Würfelspiel* is typically arranged arbitrarily so that it is not at all clear that the choices for each measure have the same general musical function. These apparently random arrangements no doubt made such games seem all the more fantastic in the eighteenth-century parlor where they were often played.

A number of composers employed *Würfelspiel* combinatorial techniques to create large-scale works. For example, Josef Riepel (1755) developed "melodic combinations in the construction of minuets, concertos, and symphonies. Within a given model

	A	B	C	D	E	F	G	H
2	96	22	141	41	105	122	11	30
3	32	6	128	63	146	46	134	81
4	69	95	158	12	153	66	110	24
5	40	17	113	85	161	2	159	100
6	148	74	163	45	80	97	36	107
7	104	157	27	167	154	68	118	91
8	152	60	171	53	99	133	21	127
9	119	84	114	50	140	86	169	94
10	98	142	42	156	75	129	62	123
11	3	87	165	61	135	47	147	33
12	54	130	10	103	28	37	106	5

Figure 1.5
A matrix for a first phrase from a *Musikalisches Würfelspiel* attributed to Franz Josef Haydn.

he seeks to achieve optimum effects by substituting figures, phrases, and cadences'' (Ratner 1970, p. 351).

The popularity of *Musikalische Würfelspiele* was extensive during the eighteenth century, particularly in Germany. Each game was capable of producing so much new music that the ''entire population of eighteenth-century Europe, working a lifetime on these games could not exhaust the combinations'' (Ratner 1970, p. 344). The creation of *Musikalische Würfelspiele*, however, did not extend beyond the Classical period nor did the form have much serious consequence. (For more on the *Musikalisches Würfelspiel*, see Eleanor Selfridge-Field's discussion in chapter 11.)

More Recent Examples

Popular music retains many of the same notational properties of the previously discussed Baroque period figured bass and shares a similar objective for virtual music: the ability to create music in many different guises while maintaining the style intended by the composer. Most popular music notation provides only a single line and chord symbols from which performers improvise their own versions of the music within the constraints provided by the implied chords. Figure 1.7 gives an example of this. Note that popular music uses a melody rather than a bass-line and note names representing chords instead of arabic numerals for inversions. However, the same kind of recombinatory principles pertain as those in figured bass.

As with Baroque figured bass, the performer of popular music is expected to supply a large number of the actual notes for the resulting music. Performers are

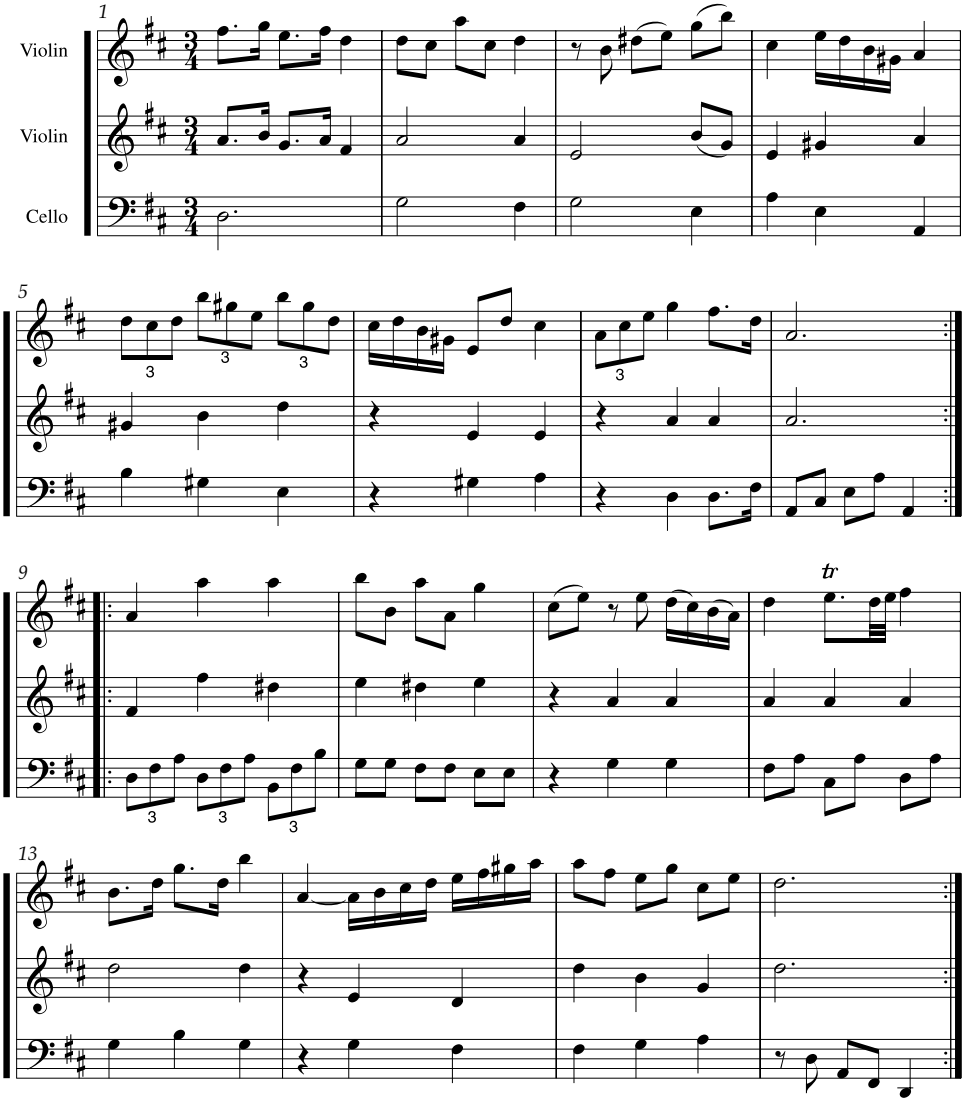


Figure 1.6
A resulting minuet derived from the *Musikalisches Würfelspiel* attributed to Franz Josef Haydn.

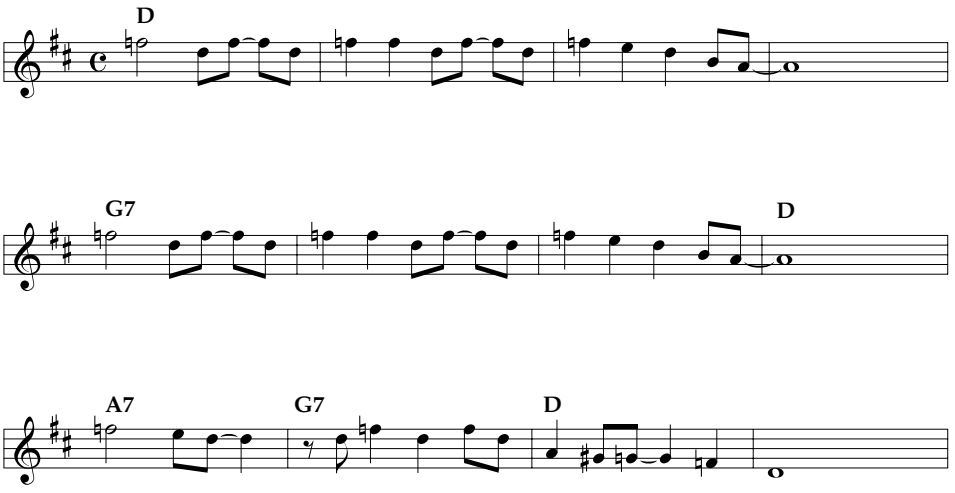


Figure 1.7
An example of popular twelve-bar blues music shorthand notation.

also expected to adhere to a logical style implied by the music as well as (often) by the title and lyrics. Thus, many different realizations can occur. Figure 1.8 presents an extremely vanilla example. Here the chords are simply iterated, much as they were in figure 1.2, the first realization of the figured bass of figure 1.1. The intended style of music barely survives this rather stagnant interpretation. On the other hand, figure 1.9 gives a much more plausible realization. Here, the left-hand figuration and the right-hand chords provide much of what audiences know as blues style. Both figures 1.8 and 1.9 are correct. The latter example, however, adheres to the style implied by the rhythm and notes of the original notation in figure 1.7.

In the Baroque figured bass and contemporary popular music we find many notational and conceptual similarities. First, both notations provide two types of information: musical notation which requires accurate performance and a shorthand for realization or improvisation. Second, both forms have constraints. In the figured bass examples, these constraints take the form of voice-leading rules and the recombination of relevant motives and musical ideas. In the popular music example, these constraints result from recombinations of possible chord notes in various registers and relevant stylistic limitations. Lastly, both examples provide performers with a fairly wide range of freedom regarding what and how many actual notes will occur and when. In short, these examples have a given part, a derived or implied rules part,

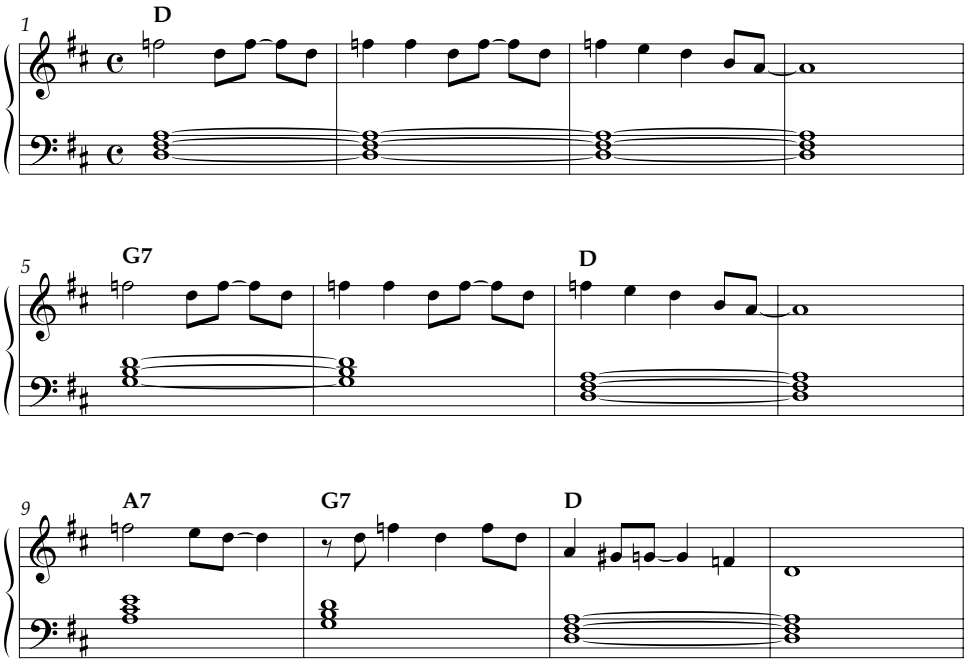


Figure 1.8
One realization of the notation in figure 1.7.

and a free part. Combined, these three elements foster the creation of innumerable style-specific realizations of the same basic given materials.

In the past fifty years or so, computers have provided the principal source for virtual music. One of the pioneers of using computers in this way was Lejaren Hiller who, in collaboration with Leonard Isaacson, wrote programs for the Illiac computer. Hiller and Isaacson's work led to the composition of the *Illiac Suite for String Quartet* in 1956 (Hiller and Isaacson 1959), one of the first such works written using computers. This innovative composition incorporates numerous experiments involving style simulation.

Iannis Xenakis uses mathematical models such as probability laws, stochastics (a mathematical theory that develops predictability from laws of probability), game theory, and Markov chains (Xenakis 1971) to compose his music. Xenakis's works often interweave his own intuitive composition with passages created by his various algorithmic computer programs which ultimately contribute to his overall musical

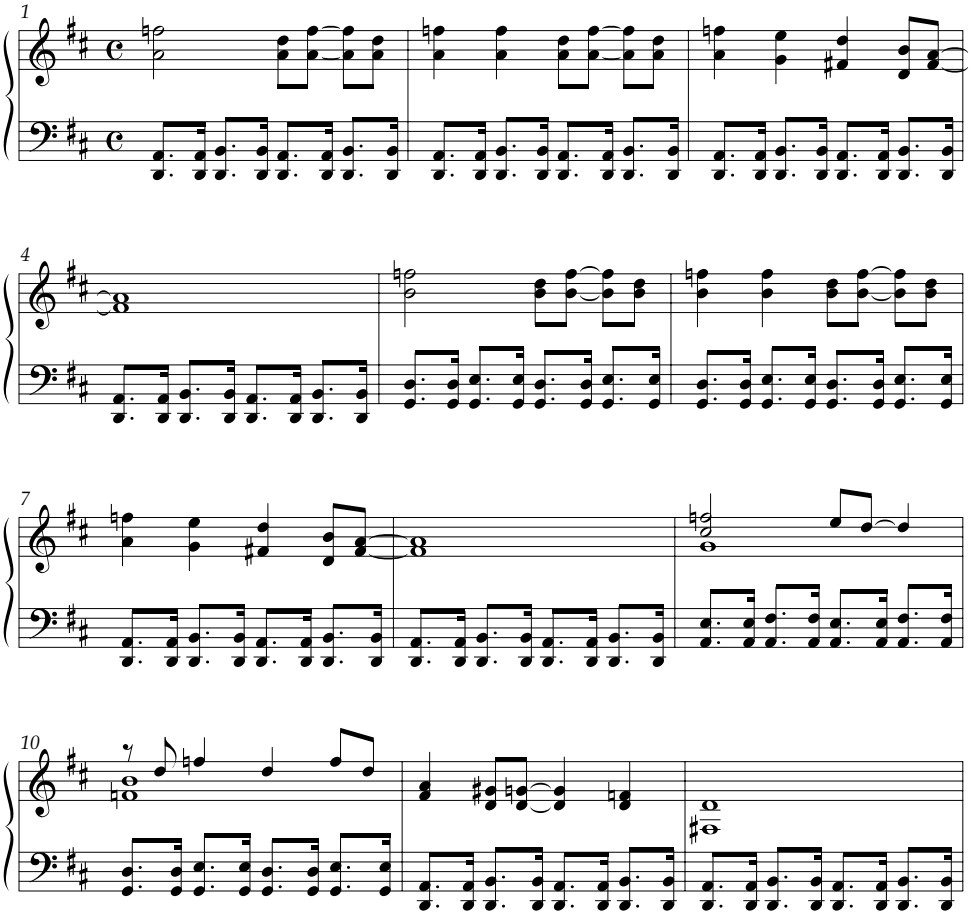


Figure 1.9
A much more stylistic realization of the notation in figure 1.7.

style. Considered by many as the progenitor of computer composition, Xenakis often alters computer-generated material to fit his musical needs.

Kemal Ebcioğlu (1987, 1992) used predicate calculus to develop more than 350 rules of voice-leading for creating chorales in the style of J. S. Bach. His program effectively portrays the basic techniques of four-part writing. William Schottstaedt created Counterpoint Solver (1989) which closely follows the exposition of species counterpoint as given by J. J. Fux around 1725. Schottstaedt's program produces logical counterpoint in a generic sixteenth-century style.

Charles Ames's Cybernetic Composer (1992) creates music in popular and jazz styles. Unlike the programs by Ebcioğlu and Schottstaedt which harmonize given melodies, Cybernetic Composer creates coherent melodies over basic chord progressions. Whether composing rock or ragtime, Cybernetic Composer often produces quite musical results. Christopher Fry's program Flavors Band (1993) produces generic jazz improvisations. Paul Hodgson's software, called Improvisor, mimics, in particular, the styles of Charlie Parker and Louis Armstrong. Improvisor composes in real time and because it mixes rhythmic and melodic patterns includes an element of improvised performance in its output.

Ulf Berggren's doctoral dissertation, *Ars Combinatoria: Algorithmic Construction of Sonata Movements by Means of Building Blocks Derived from W. A. Mozart's Piano Sonatas* (1995), takes snippets of music from sonatas by Mozart and recombines them according to what the program interprets as sensible musical orders. While the music produced often reveals both its sources and the seams by which these sources connect, the program does create occasional moments of interest. Figure 11.7 shows the opening of a first movement Mozart-like sonata as presented in Berggren's dissertation.

Christopher Yavelow's *Push Button Bach* program produces two-part inventions, arguably in the style of J. S. Bach. Figure 1.10 shows one of the works produced by this program. New output is rendered directly in music notation, one of the most attractive features of Push Button Bach. Purists will no doubt argue that this program's output falls far short of being truly Bach-like in style. Its simplicity and accessibility make it nonetheless one of the first such programs freely available over the Internet.

More recently, Dominik Hörnel and Wolfram Menzel (1998) have used neural nets to create music with stylistic similarities to composers of the Renaissance and Baroque periods, focusing primarily on harmonization and melodic variation. Their work departs from previous approaches based on programmed rules. Hörnel and Menzel provide their program with one or more examples of music which the neural network then "learns" through a process called backpropagation.

4 Composing Style-Specific Music

I began Experiments in Musical Intelligence in 1981 as an attempt to create new music in my style. I soon realized, however, that I was too close to my own music to define its style in meaningful ways, or at least in ways which could be easily coded into a computer program. I opted therefore to create a program to compose music in the style(s) of composers whose works I had studied since my early youth—the classical composers of Western Europe.

Coding the Rules of Basic Part-Writing

My initial idea for Experiments in Musical Intelligence was to code the rules of basic part-writing (how chords smoothly connect to one another) as I understood them since such processes were common practice for many classical composers. These rules resemble those typically taught to undergraduate college music students and derive from generalizations made about music of the late Baroque to the middle Romantic periods of music history—roughly 1700 to 1860. After much experimentation, I created a program which produced a kind of styleless music which basically adhered to these rules. Figure 4.1 shows an example of music composed by this program. As can be seen and heard, the results are fairly sterile and while generally correct, convey little of the vibrancy of the music from which the rules derive. The statistical abstractions of the rule-making and rule-applying processes have neutralized any real musical interest.

While some of the output from this version of the program proved relatively successful, most of the music was uninteresting and unsatisfying. As well, having an intermediary—myself—form abstract sets of rules for composition seemed artificial and unnecessarily premeditative. Coding rules for a specific composer's style also meant that I had to create new versions of the program each time I wanted it to produce music in a different style. I therefore created a new program to derive rules from music coded in a database.

Recombinancy

I began by having this new program segment Bach (1685–1750) chorales into beats and saving these beats separately in objects using object-oriented programming (OOP) techniques. Object-orientation allows storage of data and analyses of those data in ways analogous to the real world, thus making access and manipulation of those data easier and more logical. Along with each beat I had this new version of the

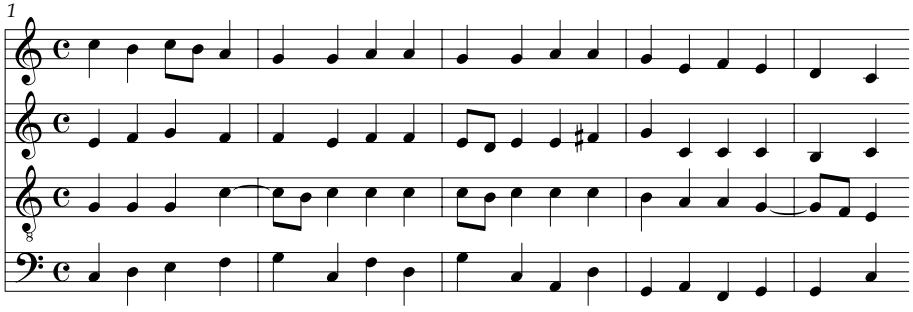


Figure 4.1
An example of music composed by an early rules-based version of Experiments in Musical Intelligence.

program retain the name of the destination pitch to which each voice moved in the next beat. I further had the program collect these beat objects in various groups called lexicons named according to the pitches and register of their entering voices (e.g., C1–G1–C2–E3, where the number suffixes reflect the octave of the note name). To compose, then, the program chooses the first beat of any chorale in its database and examines this beat's destination notes. The program then selects one of the stored beats in the lexicon whose name matches the current beat's destination notes. New choices then create the potential for different offbeat motions and different following chords while maintaining the integrity of Bach's voice-leading.

Seeing—and hearing—is believing, and following the creation of an actual new phrase of music, while to some extent repeating the process just described, can usefully establish a modus operandi for later chapters.

Figure 4.2 shows the first phrase of Bach's Chorale no. 26 (Bach 1941). Note how this music can be grouped by quarter-note beat with each voice in the groupings having a distinct destination, as just described. Figure 4.3 shows each beat of figure 4.2 boxed to show this grouping process. Figure 4.4 then provides an exploded view of the first three beats of Chorale no. 26 with lines connecting the last heard notes of one beat to the first heard notes of the next beat (destinations). These lines also follow the voice-leading rules of four-part writing, the rules which the program will attempt to emulate in creating a new chorale.

Given that databases can have thousands of chords stored in objects, many will appear more than once. Often these duplications will move in different directions, a fact that contributes to the program's ability to create new chorales. To increase the number of possibilities, Experiments in Musical Intelligence transposes all of the

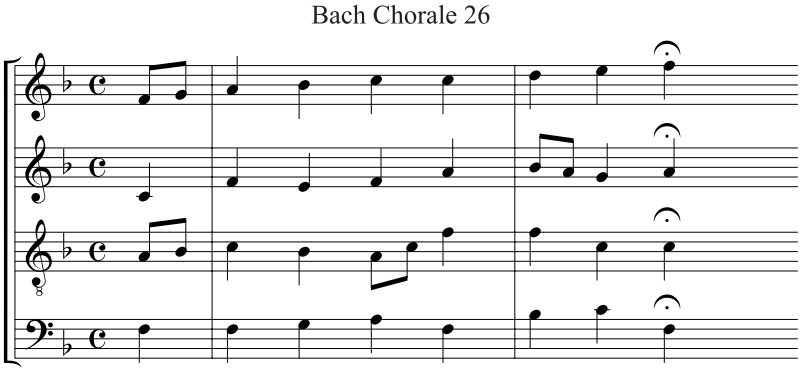


Figure 4.2
Bach Chorale no. 26, phrase 1.

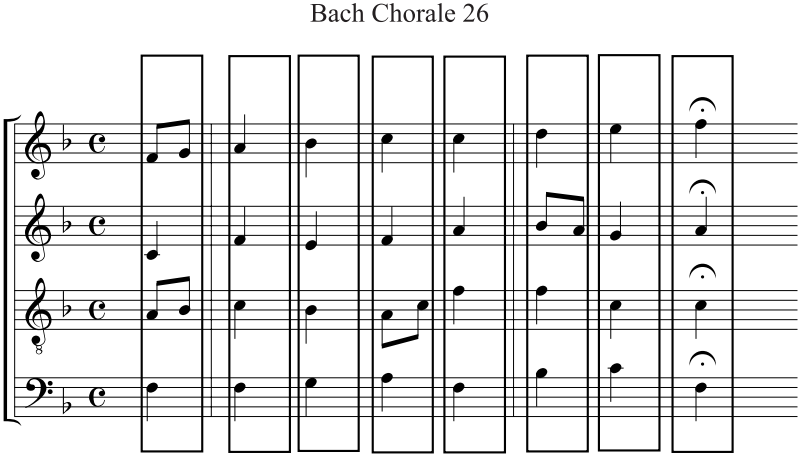


Figure 4.3
Bach Chorale no. 26, phrase 1, with beats separated.

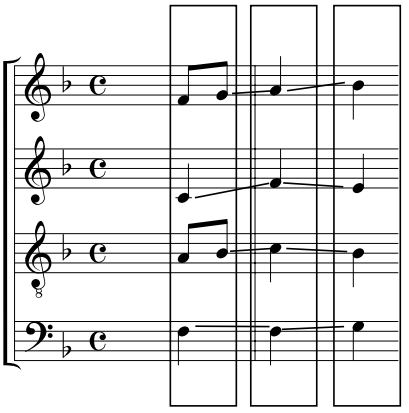


Figure 4.4
An exploded view of the first three beats of Bach Chorale no. 26, phrase 1, with destination notes highlighted by connecting lines.

music in its database by phrase to a single key before analysis. Any key will do as long as the key choice is applied equally to all phrases.

Figure 4.5 shows how the third beat of the third full measure of Bach Chorale no. 6 has the identical notes as those of the second beat of Bach Chorale no. 26. Measure numbers include pickup measures. Experiments in Musical Intelligence substitutes this new beat in place of the actual second beat of Bach Chorale no. 26, producing the beginning of a new phrase. Since the destination notes remain the same as those in the original chorale, the integrity of Bach's voice-leading remains intact while the music continues in a different way than Bach originally intended (refer to figure 4.2).

Figure 4.6 continues this process by replacing the next beat of Bach Chorale no. 6 with music from measure 14 of Bach Chorale no. 323. This choice takes the newly created music further yet from the original music of Bach Chorale no. 26. Figure 4.7 then links Bach Chorale no. 224 with Bach Chorale no. 323. The program's failure to discover substitutes for these few beats of the new chorale is due to a lack of logical substitutes in the database. The resultant phrase, however, is new and substantially different from that shown in figure 4.2. In this manner, this simple version of Experiments in Musical Intelligence creates interesting new chorales, all correct in terms of chord-to-chord rules for voice-leading, but without the program having to apply any user-supplied rules. I use the term *recombinancy* to describe this approach. The music, in effect, inherits the voice-leading rules of the works upon which it bases its replications.

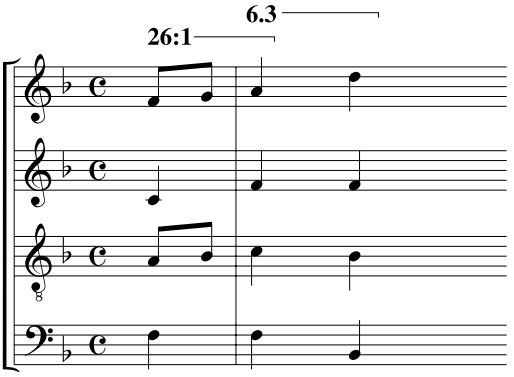


Figure 4.5
Music from Bach Chorale no. 6, m. 3 grafted to the first beat of Bach Chorale no. 26.

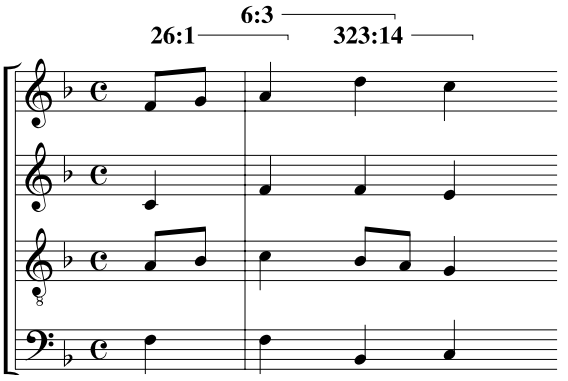


Figure 4.6
Figure 4.5 continued with music from Chorale no. 323.

Structural Logic

The problems with this approach, however, are many. First and foremost, the resulting music wanders with very unbalanced and uncharacteristic phrase lengths. This occurs because phrases cadence only when the program randomly encounters a cadence in the original music. In essence, the program has no real logic beyond the chord-to-chord syntax. Further, the program has no large-scale structure and phrases simply string together randomly, usually in a single key, and without any sense of the kinds of repetition and development necessary for intelligent composition.

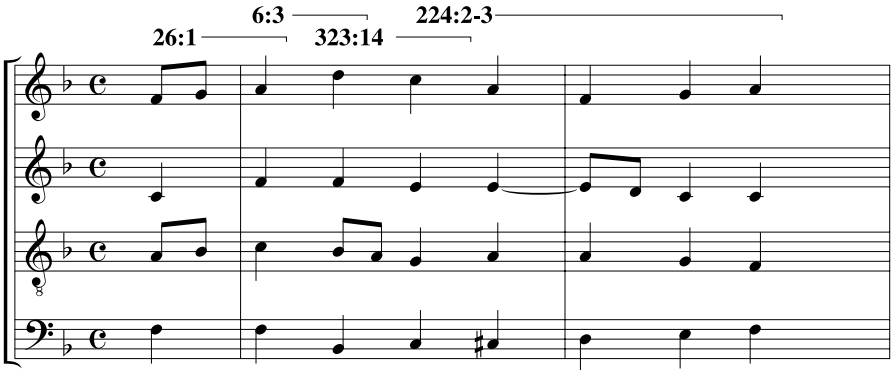


Figure 4.7
Figure 4.6 continued with music from Chorale no. 224.

In order to provide some sense of structural logic, and wanting to avoid directly programming the various constraints of musical form myself, I rewrote the program to inherit more of the structural aspects of the music in its database. This inheritance involved coding an analysis subprogram that stores other information with each beat along with its immediate destination notes. Storing references pertaining to general temporal location of cadences, for example, helped the program create effective phrase and section endings without duplicating entire phrases.

Figure 4.8 provides a visual example of how Experiments in Musical Intelligence produces music with proper phrase lengths. The program first chooses a chorale upon which to model its new composition. The program stores this single-chorale data in a separate location for consultation during the composing process. As new phrases are composed using the previously described recombination, the program maintains a connection with the structural model in regard to overall form, basing choices for new beats on how well their original position in relation to cadence relates to the current in-progress phrase's relation to cadence. The model also provides general information such as number of phrases, cadence types, and so on. The model, however, does not dictate the absolute length of phrases (determined within reasonable limits) since doing so could force dead-ends, and retaining too much information can cause the program to simply reiterate one of the chorales in its database.

Figures 4.9 through 4.13 show successive phrases completed in the same manner as the phrase shown in figure 4.7. Each of the source chorale numbers appears above the associated music. Figure 4.14 then shows how the new chorale matches its particular model, Bach Chorale no. 299. While the notes differ between the new music and the actual Bach, the functional relationships between the successions of cadences do not.

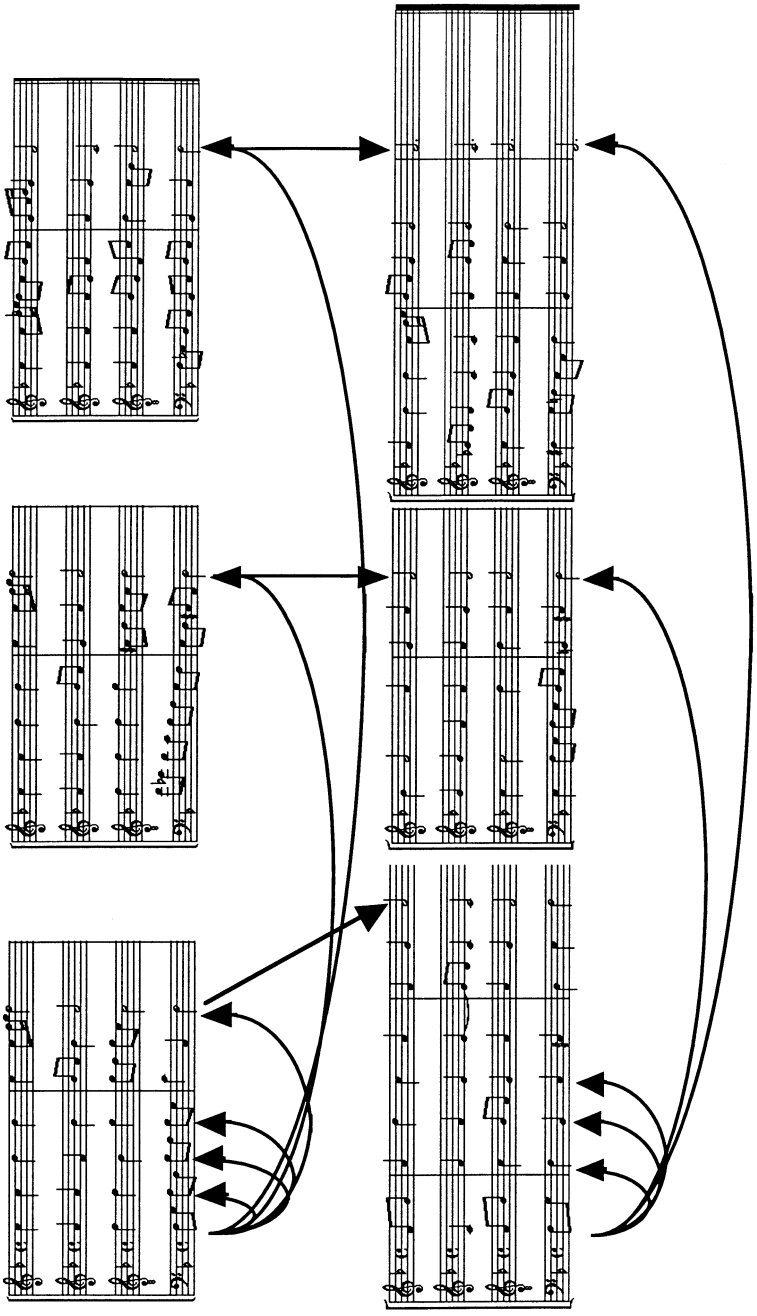


Figure 4.8
A Bach chorale (top) used as a model for a new Experiments in Musical Intelligence chorale (bottom).

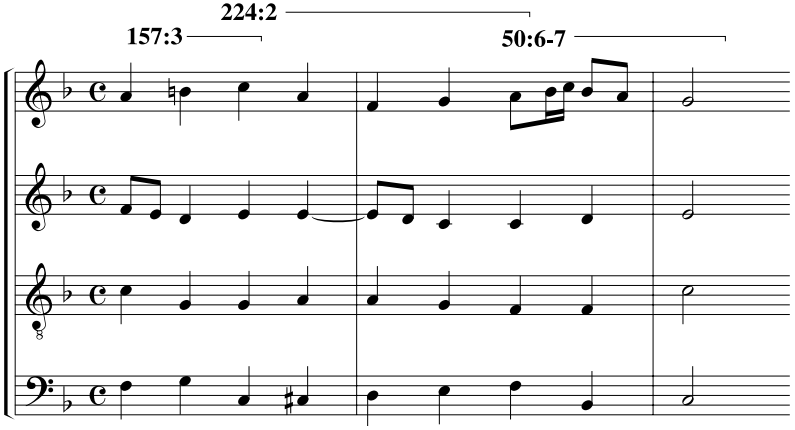


Figure 4.9
A second phrase of a new chorale.

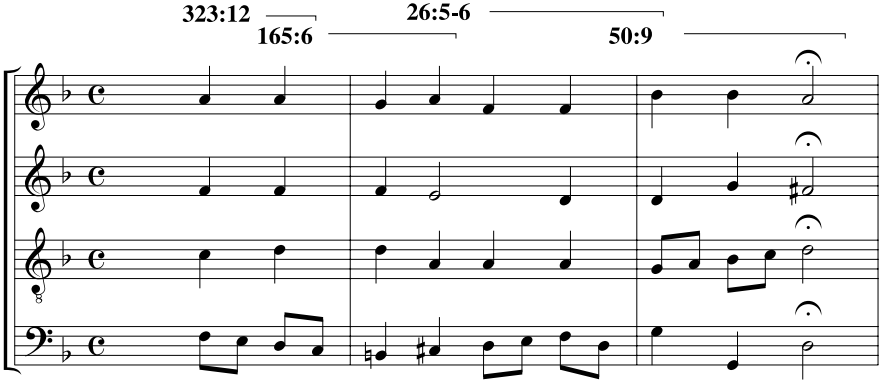


Figure 4.10
A third phrase of a new chorale.

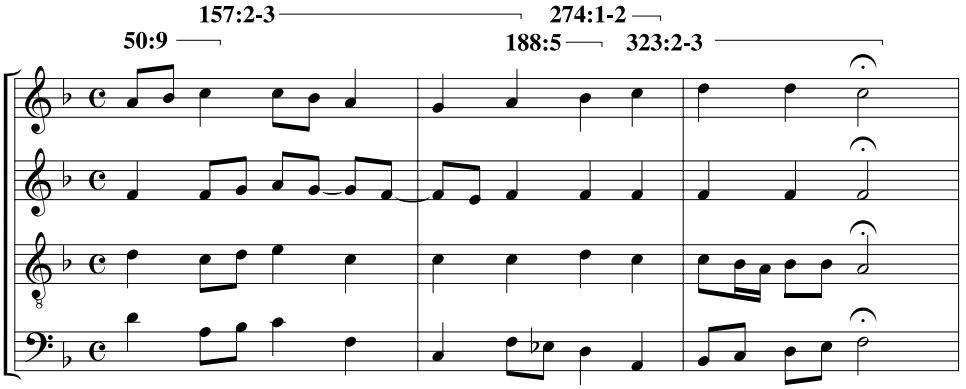


Figure 4.11
A fourth phrase of a new chorale.

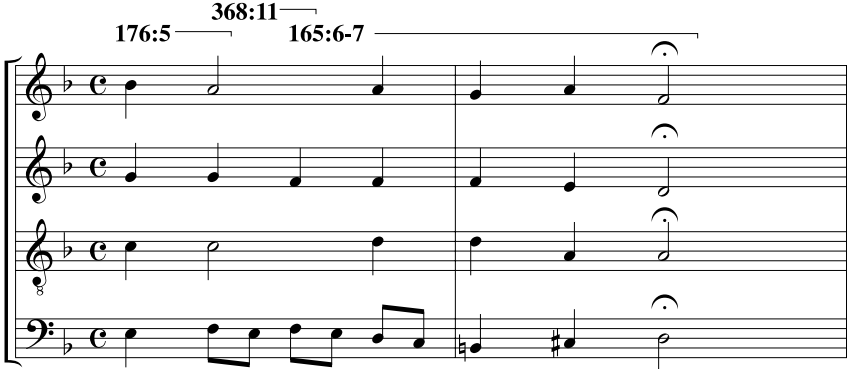


Figure 4.12
A fifth phrase of a new chorale.

Thus, while the phrases themselves accrue by successively adding material at the local level, larger structures proceed by developing a logical succession of cadences. Viewed closely, this process projects what music theorists refer to as complete authentic cadences from incomplete authentic cadences, full cadences from half-cadences, and so on (see chapter 2 of Cope 1991a for further information). The creation of movements and works follows much this same process.

Modulation presents a special case for Experiments in Musical Intelligence. When the program encounters music that modulates, it allows a new phrase to modulate as well and then transposes the ensuing phrase to the new key so that new music estab-

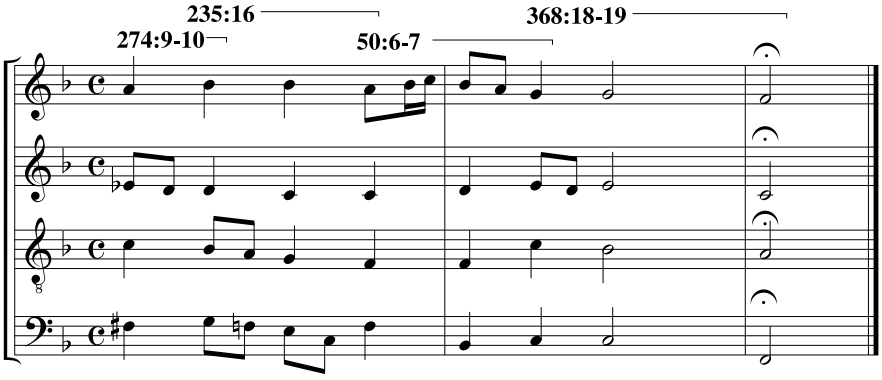


Figure 4.13
A final phrase of a new chorale.

lishes this key appropriately. Passages that modulate are then coerced back to the original keys or to new keys following the chosen structural model, as demonstrated in figure 4.14.

Since the database music has all been transposed to the same key prior to composing, the program's initial key of a newly composed chorale is that same key. However, to add variety to output, I determine the final overall key based on the range limitations of the performing voices or instruments and, as possible, other factors contributing to ease of performance.

Variations

Even with hundreds of examples of original music in a database, however, Experiments in Musical Intelligence will occasionally produce music with recognizable parts (see particularly the Mendelssohn example in appendix D). The strange juxtaposition of known materials can cause even well-formed output to be unacceptable. To alleviate these problems, the program incorporates numerous variations of the recombinant processes thus far discussed.

For example, Experiments in Musical Intelligence diatonically transposes measures to other scale degrees. Diatonic transposition differs from exact transposition because diatonic transposition produces music with altered intervals. Figure 4.15 demonstrates this process. Note that many of the actual intervals change from transposition to transposition because the notes must conform to a single key. With these added

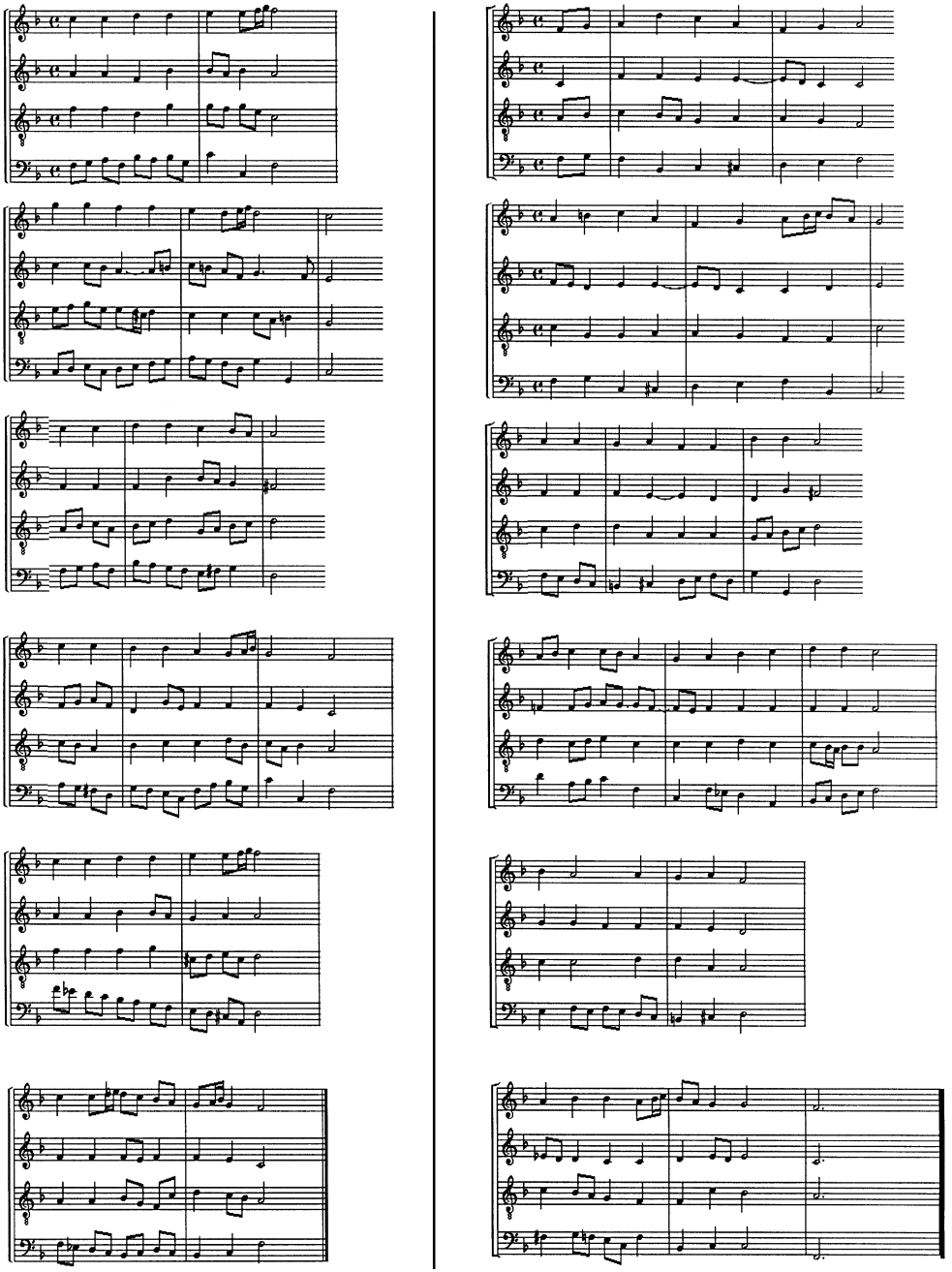
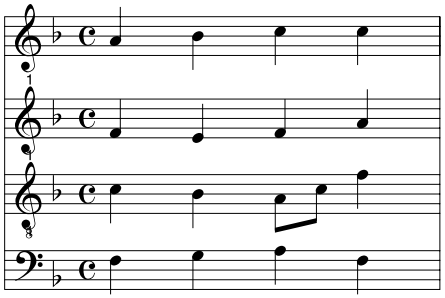
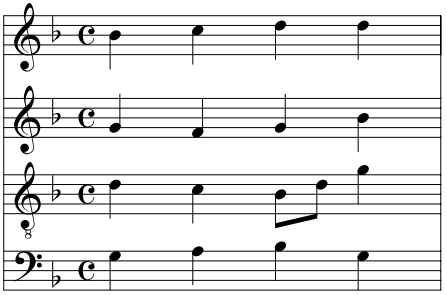


Figure 4.14
Transposed version of Bach Chorale no. 299 (the model, to the left) aligned with the newly composed Experiments in Musical Intelligence chorale (to the right).

(a)



(b)



(c)

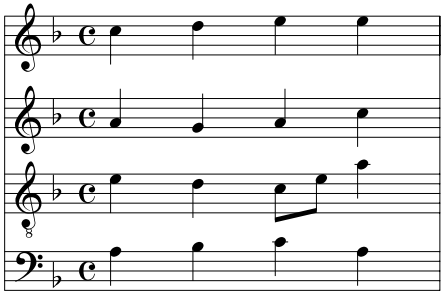


Figure 4.15
Diatonic transpositions of the first full measure of Bach Chorale no. 26.

transpositions, the program increases database size sevenfold, thus often creating new-sounding music even when it has very little data present in its database.

Experiments in Musical Intelligence also includes an elaborate subprogram capable of combinatorially reorganizing music by voice as well as by beat. I term this reorganization *transformational* recombination, since the music being recombined is transformed (Cope 1996 discusses a subset of transformative recombination called MATN). This transformation may take place between the various voices of a single beat or between voices of different beats which have the same function and entering notes. Figure 4.16 demonstrates how various recombinations—including octave transpositions—succeed in transforming beats of Bach chorales into new music.

Figure 4.16a shows vertical voice interchange. Here the voices of the same beat (the beginning of the last phrase of Chorale no. 235) are interchanged, effectively transforming the beat of music. This process poses a risk for the program in that some of the voices now have different ending notes which may not as easily connect with appropriate configurations in the database. Figure 4.16b shows examples of voice interchange between the second measures of Bach chorales nos. 6 and 26. These two chorale measures have identical initial notes. When recombined as in the three examples which follow Bach's originals, rules of counterpoint have been maintained though the resultant music differs from the original music in many ways.

Another form of transformational composition involves retaining voice order but transposing one or more voices by an octave or other interval. Such transformation requires that the transposing voice not cross other voices. As will be seen in chapters 7 through 9, this form of transformation allows some measures, otherwise not available for recombinancy, to connect properly. However, such transformational composition can be risky in that transposing by any other interval than an octave can cause serious stylistic discontinuities such as inappropriate harmonic dissonances.

Texture

All music does not follow such consistent textures as do Bach chorales. Most music in fact varies significantly in terms of numbers of voices. This texture shifting can cause numerous problems for a recombinant program like Experiments in Musical Intelligence. Likewise, beats having the same function and destination notes can have very different characters producing sudden shifts in output music. No matter how carefully works are chosen for the database, almost anything besides Bach chorales will create incongruities between individual beats during recombinancy—combining music with the proper connectivity and structural order with music of completely

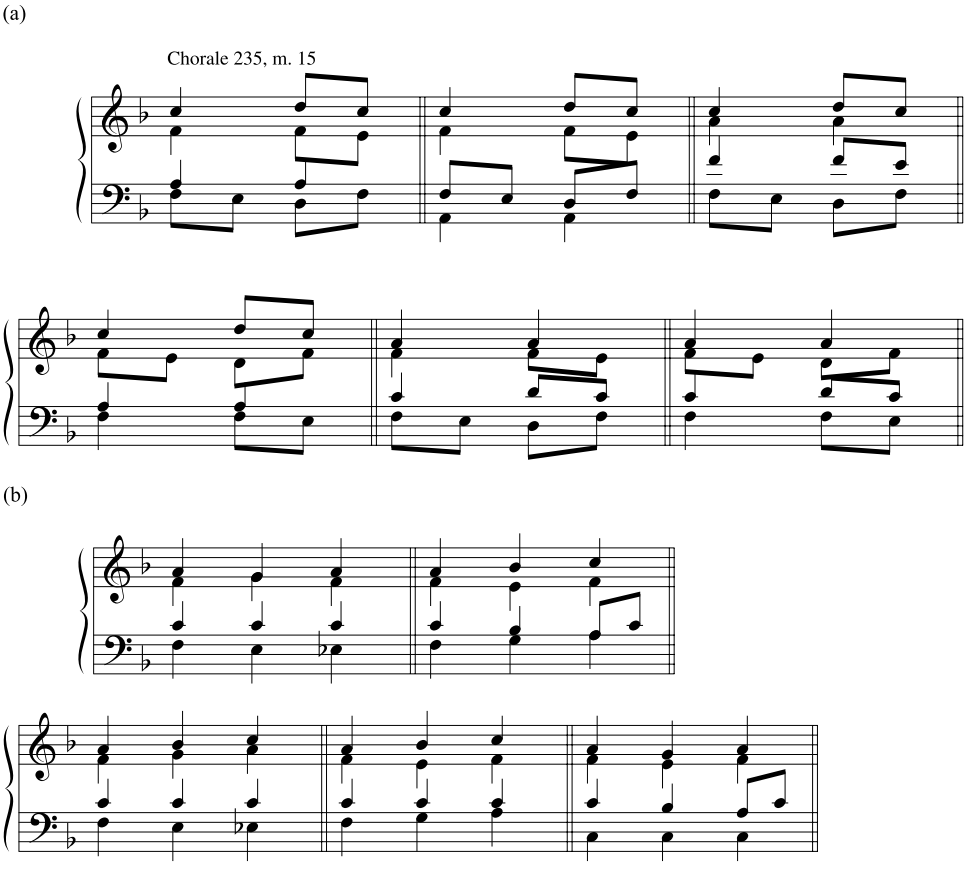


Figure 4.16
Examples of transformation: (a) vertical voice interchange of the beginning of the last phrase of Chorale no. 235; (b) examples of voice interchange between the first three beats of the second measures of Bach Chorale no. 6 and no. 26 (shown first).

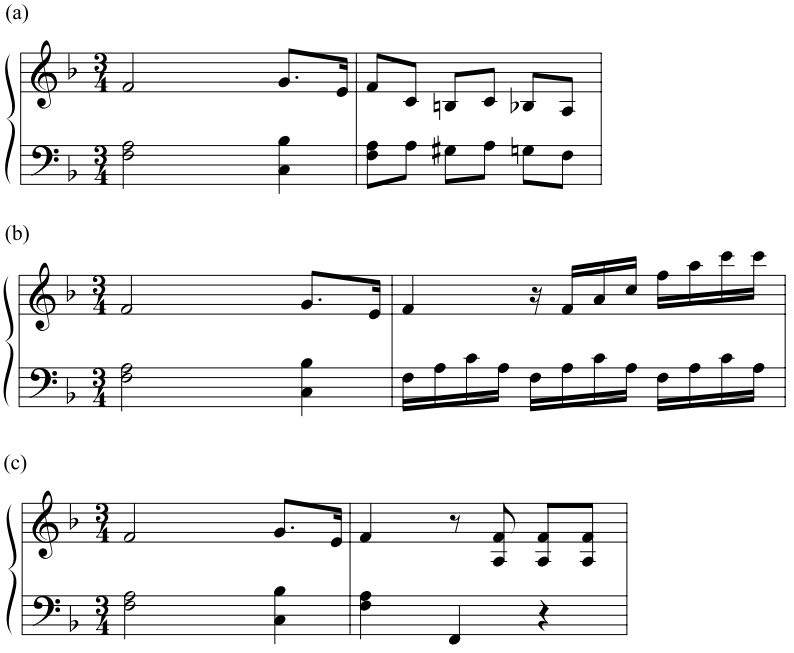


Figure 4.17
Transposed and clarified (see chapter 8) versions of (a) mm. 1–2 of Mozart's K. 284, mvt. 2; (b) juxtaposition of m. 1 of K. 284, mvt. 2 and m. 62 of K. 310, mvt. 2, which does not work; and (c) juxtaposition of m. 1 of K. 284, mvt. 2 and a slightly transformed version of m. 61 of K. 330, mvt. 2, which works effectively.

different character (see figures 4.17 and 8.1 for examples of this kind of juxtaposition). One way to avoid this problem involves storing representations of the musical texture and character of each segment so that continuity can be maintained during composition. In fact, Experiments in Musical Intelligence analyzes such continuities in the original music before it breaks the music into segments to ensure that changes of continuity take place at musically reasonable locations in the output.

To accomplish this continuity analysis I use a variety of techniques borrowed from various fields, including artificial intelligence (AI) and natural language processing (NLP) which I discuss in detail elsewhere (Cope 1991a, 1996). These techniques provide the basic logic required for the assembly of segments of music from different works by the same composer. In brief, this means that only segments of compatible music are chosen from the available segments with identical initial notes. This compatibility results from storing abstractions of each segment's musical texture and character and from mapping these to the model composition in the database.

Figure 4.17 provides a simple demonstration of how correct connections in terms of destination notes can still produce disjunct music in varying textures. Note here that only the melodic destination is considered important—harmonic connectivity in most styles need not be so carefully monitored as in Bach chorales. Likewise, recombinancy by measure, rather than by beat, works effectively in this music. Figure 4.17a shows two original measures by Mozart, the first measure of which will become the first measure of a hypothetical new computer-composed sonata movement. Figure 4.17b provides an example of how, even when the destination note of the melody matches, the characters of the two groupings can conflict and create contrast where none exists in the original. The sudden sixteenth-note motion in the second measure here dramatically shifts the character of the music. Figure 4.17c, on the other hand, demonstrates how, when the program considers the contextual continuity of the characters of the first and subsequent measures, a more logical second measure results.