# Automated Composition in Retrospect: 1956–1986

Charles Ames

**Abstract—This article chronicles computer programs for automated musical composition over the last 30 years, providing musical examples from 11 computer-composed works. The author begins by focusing on early American efforts to impose stylistic controls through random sampling and testing, then describes European statistical approaches used until 1970. The rise of 'interactive' compositional utilities during the 1970s is linked to the growing proliferation of on-line computers. The current decade has been a period of eclectic interests: the resurgence of statistical procedures, the introduction of 'top-down' recursive grammars, the adaptation of problem-solving techniques from Artificial Intelligence and the continuation of interactive efforts.**

## I. INTRODUCTION

This article chronicles efforts to harness the computer as a tool for compositional decision making, beginning in the mid-1950s and working up to the present decade. Although precedents for automated composition exist throughout musical history, never have such efforts posed as great a challenge to what for many have been fundamental assumptions about the nature of creativity and the aesthetic purposes of composition. It is therefore not surprising that these developments have met with continuing—and often virulent—resistance. Too often, however, the limitations of one or two individual approaches have been extrapolated into conclusions that computers are inherently unsuited to compositional problem solving. Even more unfortunate has been the tendency among many critics to base judgments on *ad hoc* aesthetic criteria that are irrelevant to what practitioners of automated composition have actually been trying to achieve.

A major obstacle to any informed understanding of automated composition has been the lack of accessible knowledge concerning how these composers have gone about realizing their inspirations as pieces of music. For most readers, only the well-publicized achievements of Lejaren Hiller [1–5] and Iannis Xenakis [6, 7] come to mind; in fact, there have been a number of other composers whose work has been equally deserving of recognition. Although Hiller made some

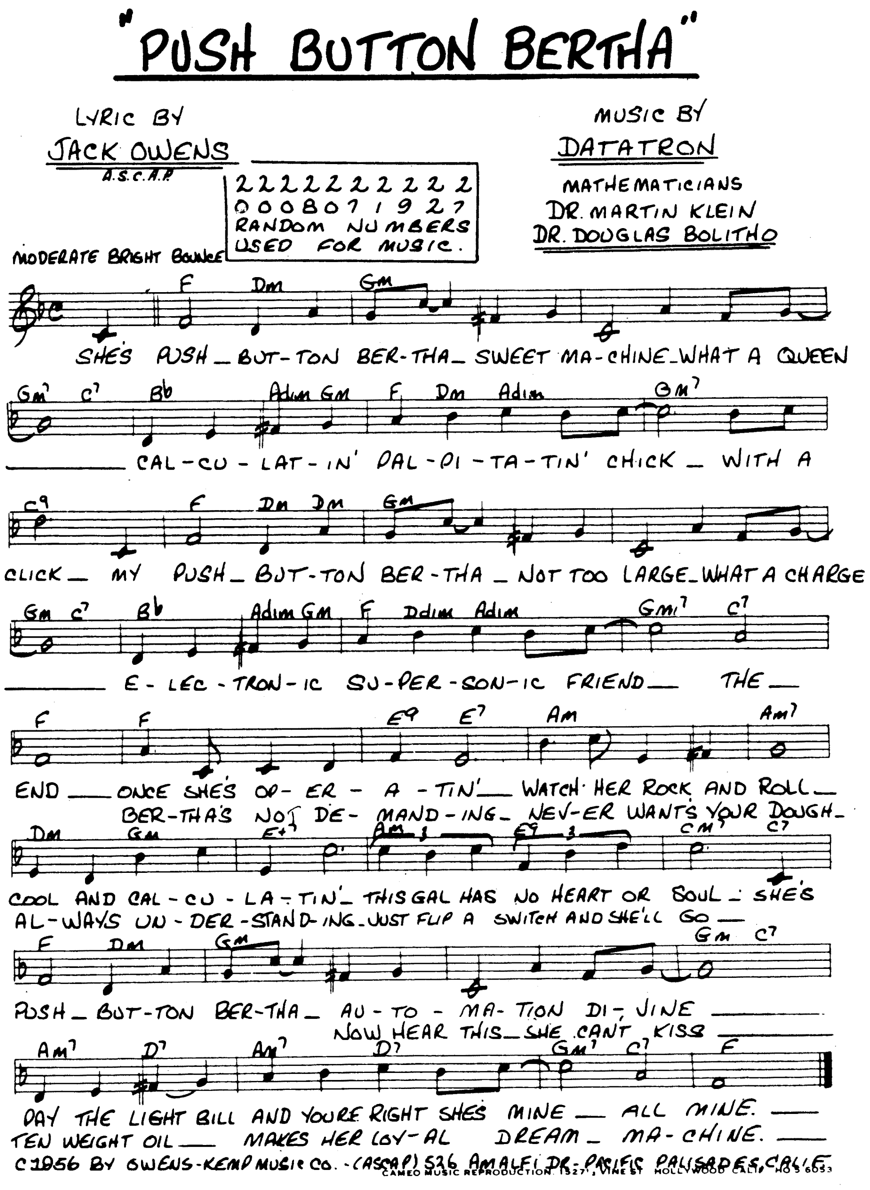
"PUSH BUTTON BERTHA"
LYRIC BY
JACK OWENS
A.S.C.A.P
MUSIC BY
DATATRON
MATHEMATICIANS
DR. MARTIN KLEIN
DR. DOUGLAS BOLITHO
2222222222
0008071927
RANDOM NUMBERS USED FOR MUSIC.
MODERATE BRIGHT BOUNCE
SHES PUSH_BUT-TON BER-THA_ SWEET MA-CHINE_WHAT A QUEEN
CAL-CU-LAT-IN' PAL-PI-TA-TIN' CHICK_ WITH A
CLICK_ MY PUSH_BUT-TON BER-THA_ NOT TOO LARGE_WHAT A CHARGE
E-LEC-TRON-IC SU-PER-SON-IC FRIEND_ THE_
END_ONCE SHES OP-ER-A-TIN'_ WATCH HER ROCK AND ROLL_
BER-THA'S NOT DE-MAND-ING_ NEV-ER WANTS YOUR DOUGH_
COOL AND CAL-CU-LA-TIN'_ THIS GAL HAS NO HEART OR SOUL_ SHES
AL-WAYS UN-DER-STAND-ING_JUST FLIP A SWITCH AND SHE'LL GO_
PUSH_BUT-TON BER-THA_ AU-TO-MA-TION DI-VINE
NOW HEAR THIS_SHE CANT KISS
PAY THE LIGHT BILL AND YOURE RIGHT SHES MINE_ ALL MINE.
TEN WEIGHT OIL_ MAKES HER LOY-AL DREAM_ MA-CHINE.
C1956 BY OWENS-KEMP MUSIC CO. -(ASCAP) 536 AMALFI DR. PACIFIC PALISADES, CALIF.

**Fig. 1. Martin Klein and Douglas Bolitho, "Push Button Bertha", 1956. Public domain.**

Charles Ames (composer), 49-B Yale Avenue, Eggertsville, NY 14226, U.S.A.

Manuscript solicited by Anthony Hill.
Received 17 January 1986.

progress in correcting this deficiency through his 1970 comprehensive survey of early efforts [8], he was able to provide no musical examples and only the barest details about how individual pieces were made. The present article seeks not only to bring the history of automated composition up to date, but also to clarify (1) what has motivated composers over the last 30 years to delegate their creative decisions to a machine, and (2) how these composers have gone about programming the machine to give them what they wanted.

## II. "PUSH BUTTON BERTHA": 1956

One of the earliest instances of automated composition was a program for composing 'Tin Pan Alley' melodies, which was created by Martin Klein and Douglas Bolitho of Burroughs, Inc. using a computer called DATATRON. An anonymously written Burroughs publication outlines the operation of the program as follows:

> The operator inspires DATATRON by first keying in a 10-digit random number. This causes the machine to generate and store 1000 single digits, each representing one of the eight diatonic notes in the scale with two allowable accidentals. The program then motivates DATATRON to pick successive notes at random, testing each for melodic acceptability as it goes along [9].

One result of this process was the melody "Push Button Bertha" (see Fig. 1), which was first aired on July 15, 1956 [10].

## III. THE URBANA SCHOOL: 1957 TO 1966

Klein and Bolitho's method of random sampling and testing provided what seemed at the time a viable emulation by computer of traditional compositional decision making. In fact, the same method had been developed independently by Lejaren Hiller and Leonard Isaacson at the University of Illinois, Urbana/Champaign. The publicity generated by Hiller and Isaacson's *Illiac Suite* [11], supplemented by the establishment under Hiller of one of the nation's earliest electronic music studios, attracted a number of individuals interested in merging the new music with the new technology. Along with Hiller and Isaacson, these people included Robert Baker [12], James Tenney [13], Herbert Brün [14–17] and John Myhill [18–20].

***Illiac Suite* Experiments Summarized**

*Experiment One: Monody, two-part, and four-part writing*

A limited selection of first-species counterpoint rules used for controlling the musical output
- (*a*) Monody: *cantus firmi* 3 to 12 notes in length
- (*b*) Two-part *cantus firmus* settings 3 to 12 notes in length
- (*c*) Four-part *cantus firmus* settings 3 to 12 notes in length

*Experiment Two: Four-part first-species counterpoint*

Counterpoint rules were added successively to random white-note music as follows:
- (*a*) Random white-note music
- (*b*) Skip-stepwise rule; no more than one successive repeat
- (*c*) Opening C chord; *cantus firmus* begins and ends on C; cadence on C; B–F tritone only in $VII_6$ chord; tritone resolves to C–E
- (*d*) Octave-range rule
- (*e*) Consonant harmonies only except for $^{6}_{4}$ chords
- (*f*) Dissonant melodic intervals (seconds, sevenths, tritones) forbidden
- (*g*) No parallel unisons, octaves, fifths
- (*h*) No parallel fourths, no $^{6}_{4}$ chords, no repeat of climax in highest voice

*Experiment Three: Experimental music*

Rhythm, dynamics, playing instructions, and simple chromatic writing
- (*a*) Basic rhythm, dynamics, and playing-instructions code
- (*b*) Random chromatic music
- (*c*) Random chromatic music combined with modified rhythm, dynamics, and playing-instructions code
- (*d*) Chromatic music controlled by an octave-range rule, a tritone-resolution rule, and a skip-stepwise rule
- (*e*) Controlled chromatic music combined with modified rhythm, dynamics, and playing-instructions code
- (*f*) Interval rows, tone rows, and restricted tone rows

*Experiment Four: Markoff chain music*
- (*a*) Variation of zeroth-order harmonic probability function from complete tonal restriction to "average" distribution
- (*b*) Variation of zeroth-order harmonic probability function from random to "average" distribution
- (*c*) Zeroth-order harmonic and proximity probability functions and functions combined additively
- (*d*) First-order harmonic and proximity probability functions and functions combined additively
- (*e*) Zeroth-order harmonic and proximity functions on strong and weak beats, respectively, and vice-versa
- (*f*) First-order harmonic and proximity functions on strong and weak beats, respectively, and vice-versa
- (*g*) *i*th-order harmonic function on strong beats, first-order proximity function on weak beats; extended cadence; simple closed form

**Fig. 2. *Illiac Suite* Experiments Summarized. Reproduced from Hiller and Isaacson [5]. Copyright 1959 McGraw-Hill Book Company. Used by permission of the publisher.**

### Hiller, Isaacson and Baker

During the same years in which Klein and Bolitho were programming DATATRON, Hiller and Isaacson were undertaking a series of compositional "experiments"* using the ILLIAC computer, which had been designed and built at Urbana. Many of their results were presented in a well-publicized concert which, coincidentally, also occurred during July 1956. Subsequently, these and later experiments were collected into an *Illiac Suite* for string quartet. Figure 2 details Hiller and Isaacson's procedures, which utilized two basic approaches:

1. Random selection constrained by lists of rules, an approach resembling that of Klein and Bolitho (Experiments 1–3).
2. Markov chains, also random, in which the relative likelihood of each option was conditioned by one or more immediately preceding choices (Experiment 4).

An excerpt from the *Illiac Suite* appears in Fig. 3.

*Enclosure of terms in double quotes indicates coined or otherwise idosyncratic terminology drawn from specific sources.

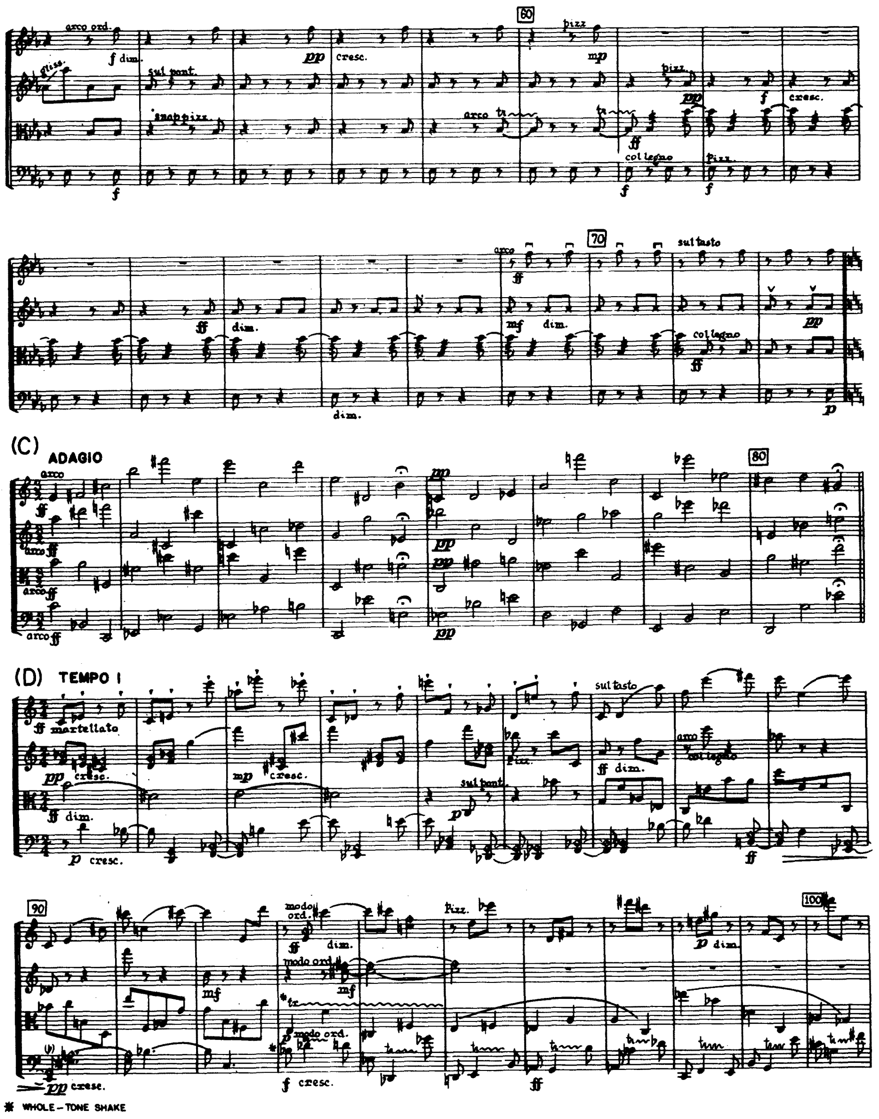

Fig. 3. Lejaren Hiller and Leonard Isaacson, *Illiac Suite*, 1957, excerpt from Experiment 3. Measures 55–72 show "basic rhythm, dynamics, and playing-instructions code"; measures 73–80 show "random chromatic music"; measures 81–100 show "random chromatic music combined with modified rhythm, dynamics, and playing-instructions code". Copyright 1957 New Music Edition. Used by permission of the publisher, Theodore Presser Company, Bryn Mawr, PA 19010, U.S.A.

A second early collaborator of Hiller's was Robert Baker, who conceived the composing utility called MUSICOMP (MUsic Simulator Interpreter for COMpositional Procedures). MUSICOMP greatly facilitated the process of developing new composing programs by managing libraries of compositional subroutines and other programming modules which individual composers could link together in a main program designed to meet their own idiosyncratic purposes. Hiller and Baker first utilized MUSICOMP to create their 1962 *Computer Cantata* [21], which employed serial methods drawn "almost *in toto*" from Pierre Boulez's *Structures* for two pianos [22] in addition to the methods employed to create the *Illiac Suite*.

Among Hiller and his colleagues, a strong motivation for automating the compositional process has been the insight into creative activity that interaction with a computer can provide. For example, Hiller and Isaacson [23] describe the *Illiac Suite* as a study of "those aspects of the process of composition . . ." which can be modeled " . . . by applying certain mathematical operations deriving from probability theory and certain general principles of analysis incorporated in a [then] new theory of communication called *information theory*". Hiller and Baker's article "*Computer Cantata*: A Study in Compositional Method" expresses a similar attitude:

> Since our primary purpose was to demonstrate the flexibility and generality of MUSICOMP, the *Computer Cantata* presents a rather wide variety of compositional procedures, some of which proved of greater esthetic value than others, and many of which could be improved [!] by more sophisticated logic [24].

## James Tenney

James Tenney had been drawn to Urbana as a graduate student by Hiller's groundbreaking course in electronic music. While at Urbana, Tenney pursued studies in computer composition. In 1961 he was invited to Bell Telephone Laboratories [25] where Max Mathews [26–28] was in the process of developing his now well-known digital sound-synthesis programs. Though unaware of the "stochastic music program" that Iannis Xenakis was developing roughly at the same time [29], Tenney had been inspired by Xenakis's statistical scores such as *Pithoprakta* and *Achorripsis*, which had been described in articles by Xenakis [30]. Tenney's programs differ substantially from Xenakis's in Tenney's use of (1) segmented line graphs controlling how compositional parameters evolve over time, and (2) hierarchic procedures devised in response to the Gestalt psychology of Max Wertheimer.

Most of Tenney's composing programs generated files of numeric data which were then realized digitally by Mathews's MUSIC4 program. However, one exception admits direct visual examination of the score: Tenney's 1963 *Stochastic String Quartet*. This piece divides into three sections, which at the indicated tempi last 50, 100 and 40 seconds. Sections divide into intermediate structural units, which Tenney calls "clangs", and these in turn divide into notes. The profile of each section is controlled by graphs of musical attributes, such as clang durations, average periods between attacks, ranges of pitches, dynamics and several aspects of timbre: vibrato, tremolo, "spectrum" (sul tasto, ord., sul pont.) and "envelope" (pizz., arco-staccato, arco-legato, arco-marcato, arco-sforzando). Each graph supplies a mean value to one or more random generators in order to select attributes for a specific clang; in turn, the program feeds these clang attributes into further random generators in order to select attributes for a note. The opening of the *Stochastic String Quartet* appears in Fig. 4.

An especially significant feature of the quartet is Tenney's elaborate procedure for notating rhythm. Not content to approximate his rhythms as displacements relative to a simple meter such as 4/4, Tenney exploits bracketed rhythmic proportions directly as a compositional resource. His notational procedure works as follows. At the broadest level, the program selects the length of each clang in quarter notes. Within a clang, it then selects an independent "gruppetto" unit for each instrumental part. Measure 13, for example, constitutes one clang lasting

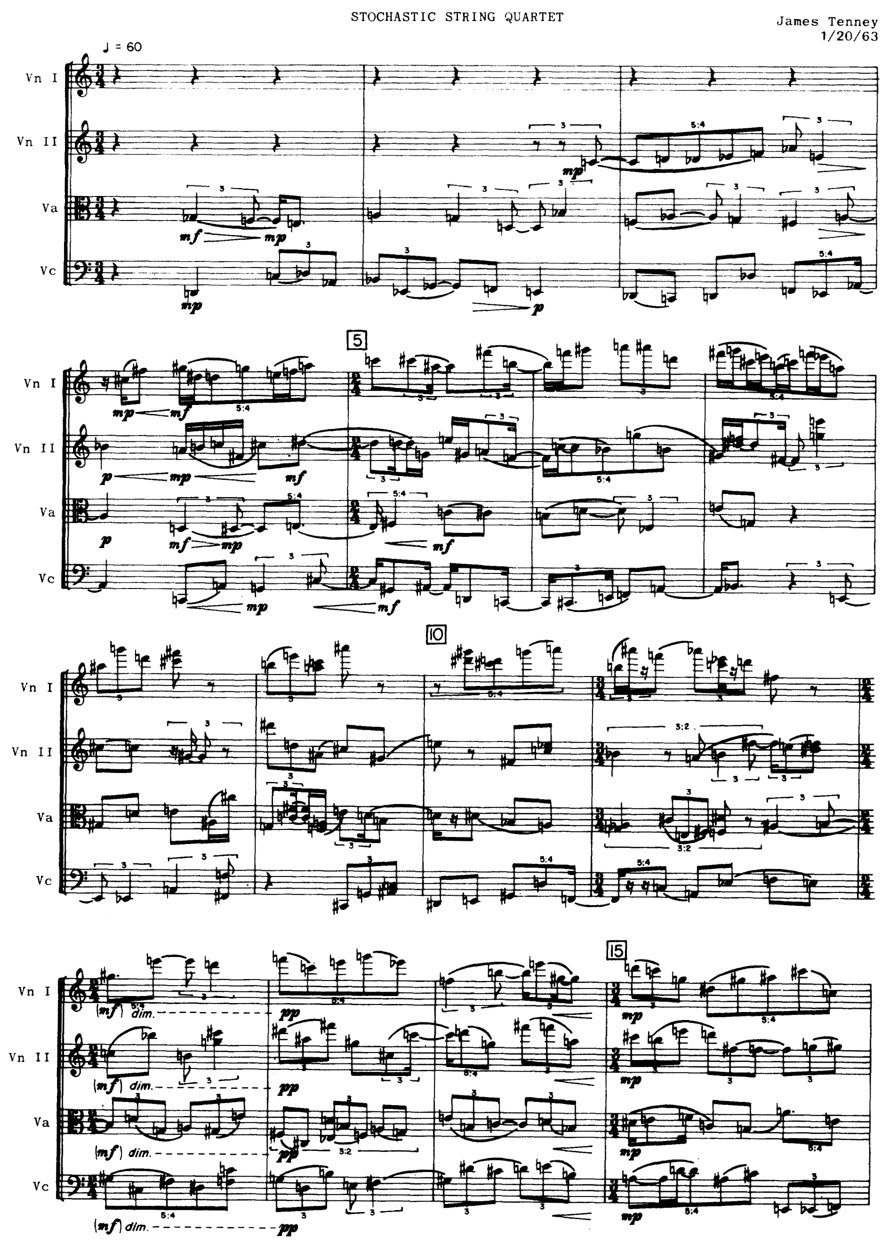

**Fig. 4. James Tenney, *Stochastic String Quartet*, 1963, measures 1–15. Copyright 1985 Sonic Art Editions, 2617 Gwynndale Avenue, Baltimore, MD 21207, U.S.A. Used by permission of the publisher.**

two quarters; the gruppetto units are 5 (violin 1), 2 (violin 2), 3 (viola) and 2 ('cello). Each gruppetto unit in turn divides randomly into smaller divisions. This continually fluid metrical structure then provides a notational framework for the 'floating-point' durations generated by Tenney's random generators.

## Herbert Brün

A member of the Urbana composition faculty from 1963 to the present, Herbert Brün employed MUSICOMP to create a number of works including his 1964 *Sonoriferous Loops* and 1966 *Non-Sequitur VI*. Like Hiller, Isaacson and Baker, Brün valued composing programs as truly empirical means for testing formulations of compositional procedures:

. . . whereas the human mind, conscious of its conceived purpose, approaches even an artificial system with a selective attitude and so becomes aware of only the preconceived implications of the system, the computers would show the total of the available content. Revealing far more than only the tendencies of the human mind, this nonselective picture of the mind-created system should be of significant importance [31].

Brün's *Sonoriferous Loops* adopts the model of Varèse's *Déserts* by alternating four digitally synthesized interludes with five short movements for live ensemble. The live ensemble employs the five instrumental parts shown in Fig. 5—flute, trumpet, double bass, mallets (xylophone and marimba) and unpitched percussion—while the interludes employ three synthetic voices. According to the technical notes on *Sonoriferous Loops* compiled by Brün, his program advances from moment to moment in the score, in effect 'scanning' each instrumental (or synthetic) part to determine whether or not the part is engaged by a note or rest. If not, then the program initiates the following steps:

1. *Rhythm.* The program randomly selects "denominators" and "numerators" from 16, 6, 5 and 4; these are applied to a common multiple of 240 to produce a duration.
2. *Rest/Play Choice.* The program next decides whether to rest or to play a note. The relative activity of each part within a section of music is controlled by "rest/play probabilities"; in the opening measures, for example, flute rests 26% of the time, trumpet 32%, double bass 50%, mallets 68% and unpitched percussion 74%.
3. *Pitch.* Should the program elect to play a note in Step 2, it will next select a *chromatic degree* and a *register*. The mechanism for selecting *degrees* consults a 12-element array which the program randomly shuffles at the beginning of each 12-note cycle. Because this mechanism supplies degrees for all parts simultaneously, the composite texture obeys a uniform distribution of degrees. *Registers* are selected at random.

Once rhythms and pitches had been determined by his program, Brün manually intervened to supply tempi, dynamics and "instrumental modes (pizz., mute, stacc., etc.)".

An excerpt from another computer-composed work of Brün's for ensemble and tape, *Non-Sequitur VI*, appears as Fig. 6. Brün's comments regarding this work evoke Xenakis's investigation in *Achorripsis* [32] of the minimal conditions necessary to create a composition. However, Brün imposes the additional requirement of 'musicality' through the mechanism of random sampling and testing:

The programming of [*Non-Sequitur VI*] mainly reflects the continuous search for answers to the following: (1) What is the minimal number and power of restrictive rules that will select from random generated sequences of elements that particular variety of element-concatenations satisfying the conditions for either recognizable or stipulated 'musical' forms and events? (2) Could a combination of stochastic choice rules with heuristic, multivalent, decision

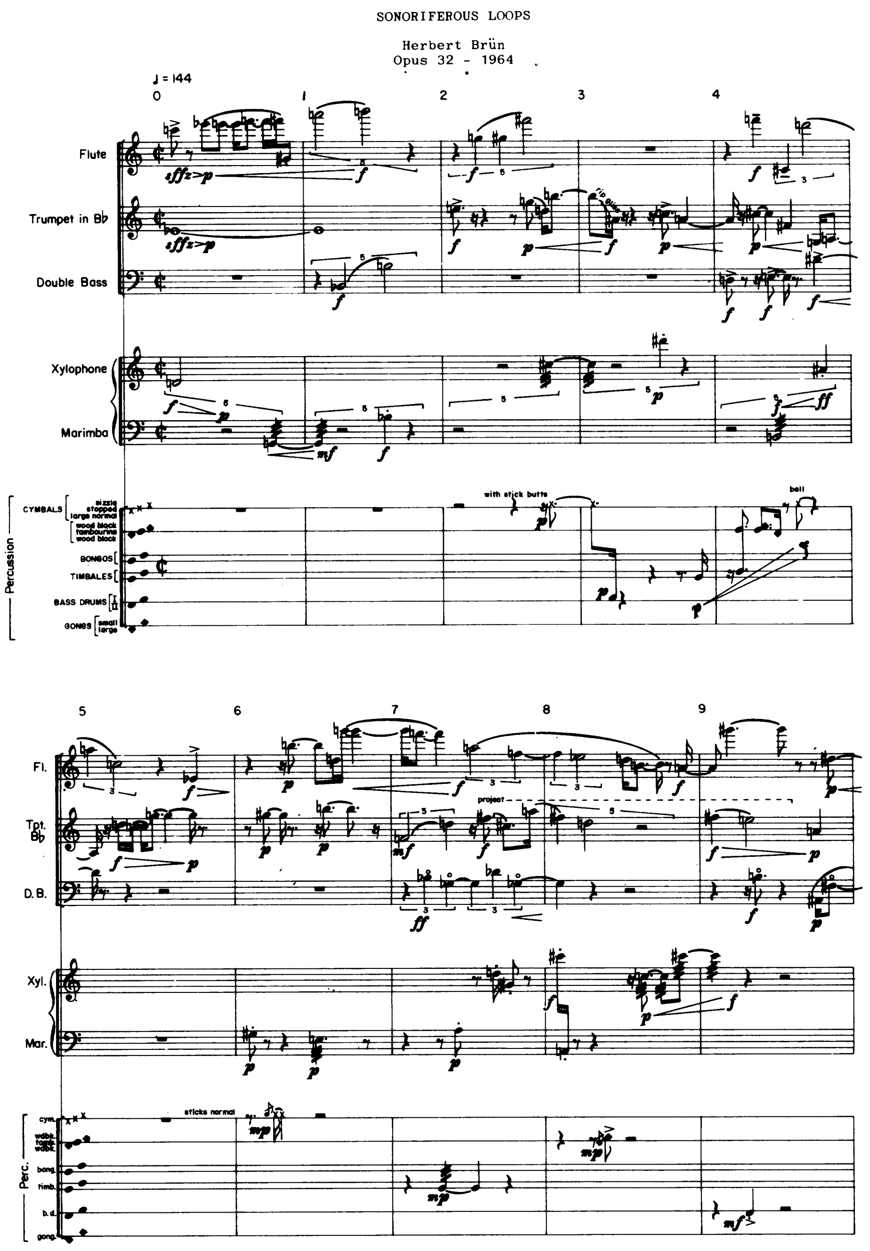

**Fig. 5. Herbert Brün, *Sonoriferous Loops*, 1964, measures 0–9. Copyright 1964 Herbert Brün. Used by permission of the composer.**

> taking procedures contribute an apparent 'musical' coherence to a chain of changes of state in a structured system? [33]

### John Myhill

Although John Myhill came to Urbana in 1964 as a professor of philosophy and mathematics, he had in fact previously undertaken formal musical studies with Michael Tippet. An avid supporter of activities then in progress at Urbana, Myhill in 1965 contributed a *Scherzo a Tre Voce* for computer-synthesized tape alone. The *Scherzo*, directly inspired by graphic methods advocated by Joseph Schillinger [34], derives melodic contours for each of its three voices—hence the title—from the eight evolving sinusoids depicted in Fig. 7.

As with Brün's *Sonoriferous Loops* program, Myhill's program scans all three voices from moment to moment; however, in this case the program determines whether or not each part is ready to begin a new contour. If ready, the program selects a shape from those given in Fig. 7, an overall duration for the new contour and one of the following three modes of performance:

1. Siren-like glissandi.
2. Discrete pitches taken from a scale relative to a rhythmic pattern: the scale degree in closest proximity to the contour at the onset of a note determines the pitch (Schillinger's approach).
3. Discrete pitches again taken from a scale. Here, each new note begins when the contour crosses the corresponding scale degree. Note durations therefore depend on the steepness of the contour as it passes from one scale step to the next.

## IV. EUROPEAN ALGORITHMIC MUSIC: 1960–1970

The first-known European composer to become involved with automated compositional procedures was Pierre Barbaud, who presented computer-composed pieces in Paris as early as 1960. Many of Barbaud's procedures are documented in a monograph entitled *Initiation à la composition musicale automatique* [35]; they include permutational methods applied to traditional harmonies, serial (12-tone) methods and methods of random selection. In his article "Algorithmic Music" he sets forth the aesthetic philosophy that musical composition "consists in creating what scholastics called an *artifactum*, that is to say something that nature does not produce". Barbaud confronts objections that automation removes the 'humanity' from composition by turning the complaint against itself:

> Music is generally called 'human' when it considers temporary or inherent tendencies of the mind, of part or all of a composer's personality. Such music is based on feeling and since it turns its back, in a sense, on pure knowledge, it might rather be called 'inhuman', for it celebrates what we have in common with all the animals rather than with what is individual to man: his reason. Algorithmic music is thus 'inhuman' only in one sense of the word, it is 'human' in as much as it is the product of rational beings [36].

Compositionally, where the Urbana school tended to focus on *local* relationships implemented in the form of stylistic rules, the Europeans emphasized *global* qualities of musical passages, as quantified by statistical distributions. Technically, it was Europeans such as Iannis Xenakis [37] and Gottfried Michael Koenig [38–42] who were the first to conceive of a composing program as a *utility* capable of generating many pieces, rather than a one-shot effort geared toward a specific compositional goal. Xenakis extolls what he considers to be the virtues of such generalized utilities in the following poetic terms:

> With the aid of electronic computers the composer becomes a sort of pilot: he presses the buttons, introduces coordinates, and supervises the controls of a cosmic vessel sailing in the space of sound, across sonic constellations and galaxies that he could formerly glimpse

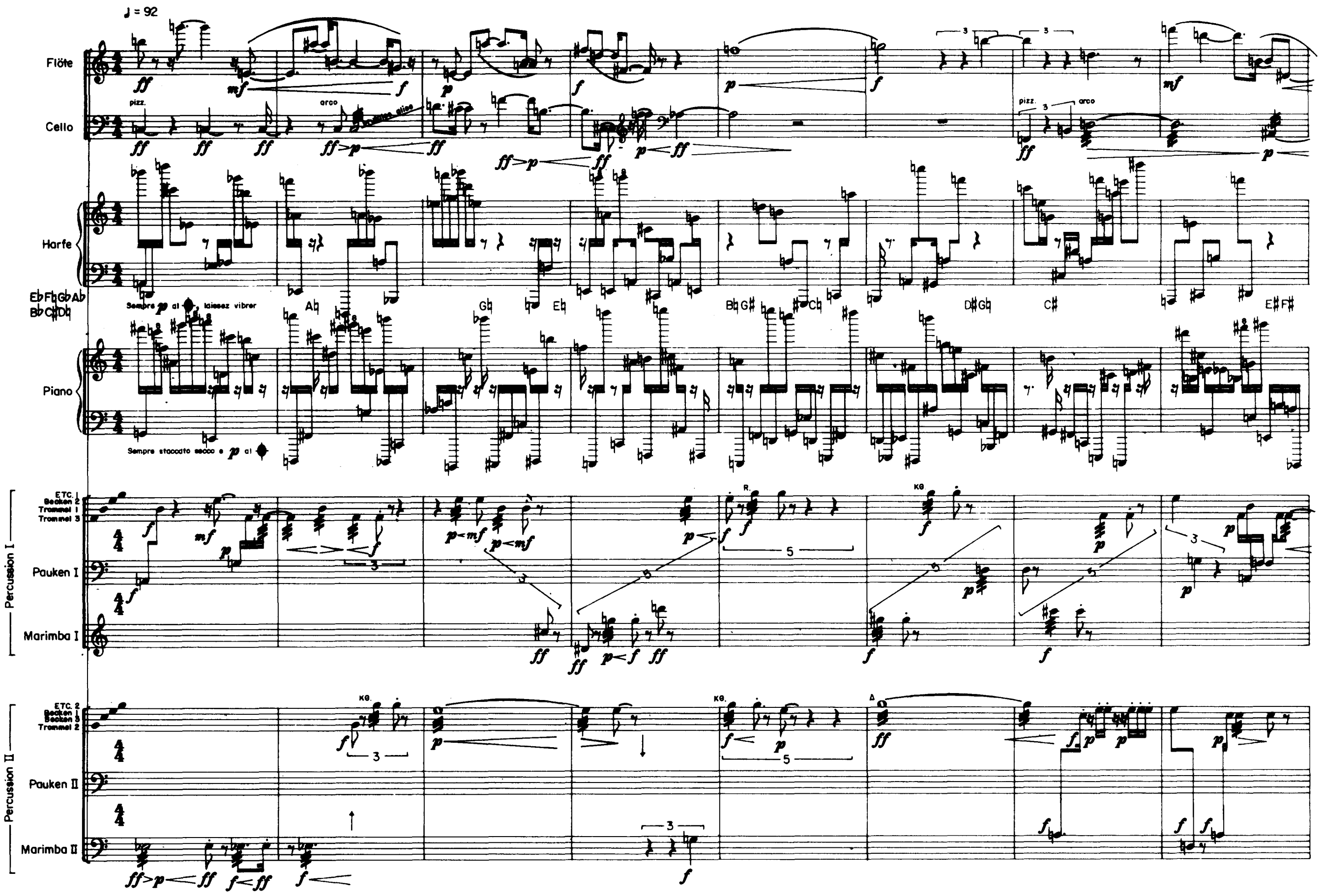

Fig. 6. Herbert Brün, *Non-Sequitur VI*, 1966, measures 1–8. Copyright 1966 Herbert Brün. Used by permission of the composer.

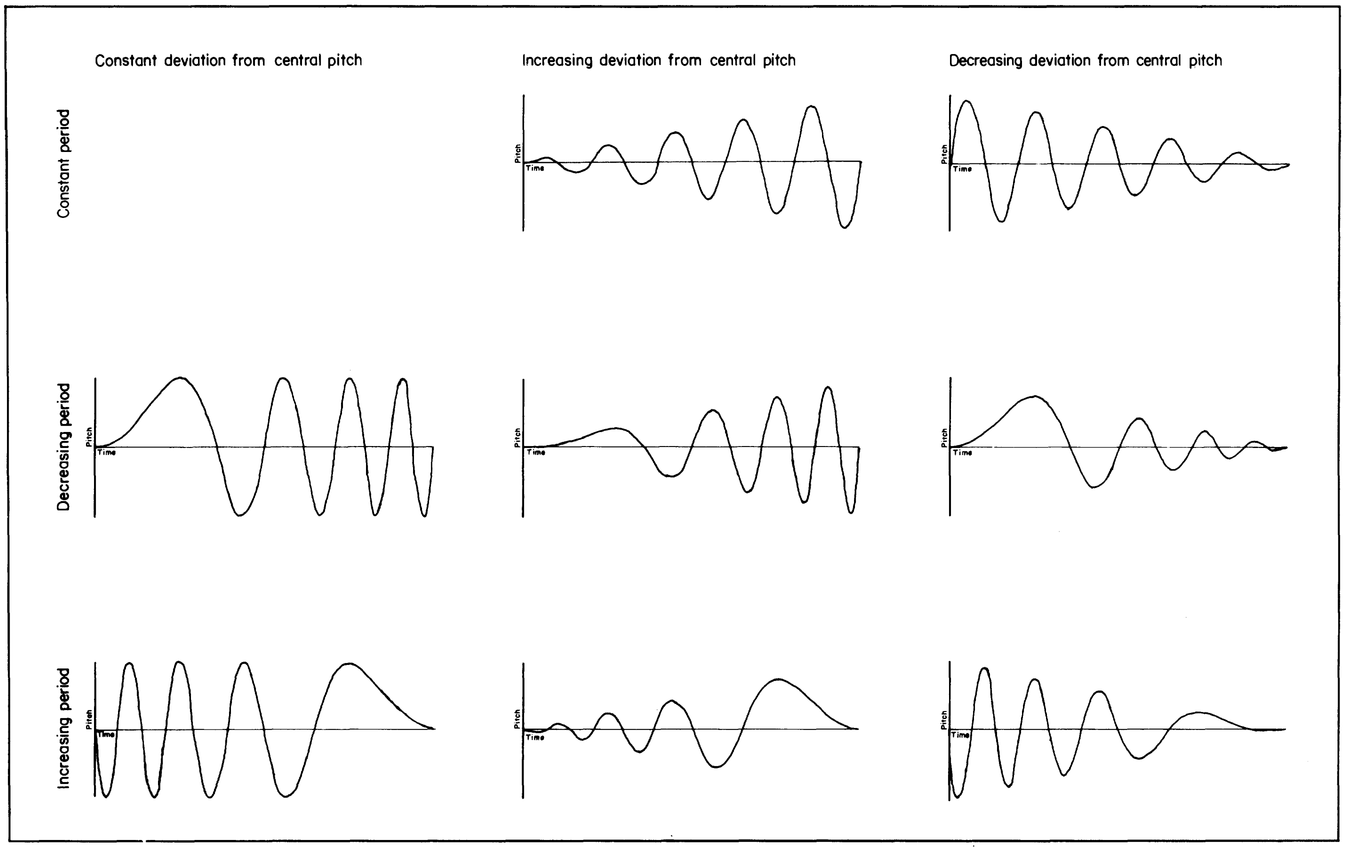
Constant deviation from central pitch
Increasing deviation from central pitch
Decreasing deviation from central pitch
Constant period
Decreasing period
Increasing period
Pitch
Time

Fig. 7. Melodic contours in John Myhill's *Scherzo a Tre Voce*, 1965. At its maximum (vertical) spread, each contour spans a one-and-one-half octave range.

only as a distant dream. Now he can explore them at his ease, seated in an armchair [43].

## Iannis Xenakis

Perhaps the best-known composer to employ computers has been Iannis Xenakis. It was Xenakis who had first introduced statistical methods of composing for live ensembles: *Pithoprakta* (1956) exploits Gaussian distribution to realize the "temperatures" of massed glissandi, while *Achorripsis* (1957) uses Poisson's distribution of rare events to organize "clouds" of sound. The aggregate effect of these statistical scores concerned Xenakis much more than specific relationships between musical elements; in his 1962 "stochastic music program" [44], he began delegating specific decisions to the random-number generator of a computer.

Following the model of *Achorripsis*, the stochastic music program employs statistical procedures to deduce a musical work with "the greatest possible *asymmetry* . . . and the *minimum of constraints, causalities, and rules*". Xenakis designed his program to create works for varied ensembles according to varied input data supplied by the composer. Among the more familiar pieces produced by Xenakis using this program are *ST/10-1,080262* for 10 instruments, *ST/48-1,240162* for orchestra (48 instruments) and *Morsima-Amorsima* ( *ST/4-1,030762*) for 4 instruments.

One can gain a sense of how Xenakis worked with his program by examining the input data used to create one such work, *Morsima-Amorsima*. This work lasts 17 minutes, 12 seconds at the indicated tempo. Its ensemble consists of violin, 'cello, contrabass and piano, and for these instruments, Xenakis established respective maximum densities at 4, 4, 3 and 15 notes per second. These values result in a combined maximum density of 26 notes per second, but Xenakis nudged this value upward slightly to 28. He also specified a combined minimum density of 0.07 notes per second—approximately one note every 13 seconds. The stochastic music program converts these "objective" limits to produce a range of "subjective" (logarithmic) densities from 0 to 6. For each of the integral points along this range of subjective densities (0, 1, . . ., 6), the program accepts a list of probabilistic weights by which notes will be drawn from each of the various *timbre classes* available from the ensemble. For *Morsima-Amorsima*, Xenakis has defined

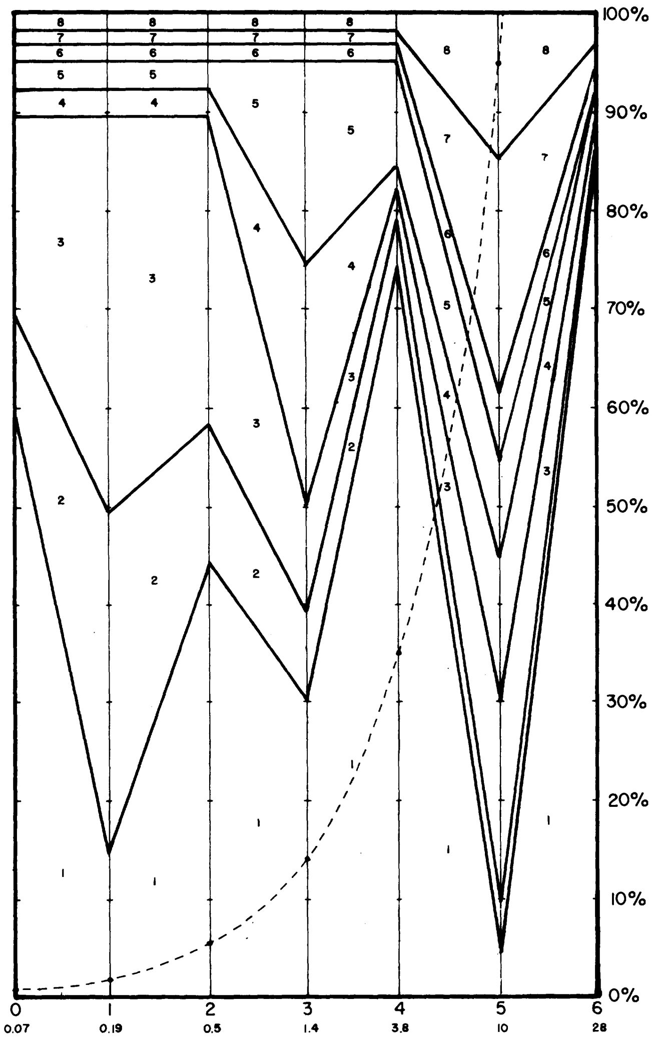

**Fig. 8. Chart of orchestration versus subjective density for Iannis Xenakis's *Morsima-Amorsima*. The indicated timbre classes are: (1) piano, (2) arco ponticello, (3) harmonics, (4) arco normale, (5) glissando, (6) tremolo arco ponticello, (7) pizzicato and (8) frappe col legno. After a diagram provided by the composer.**

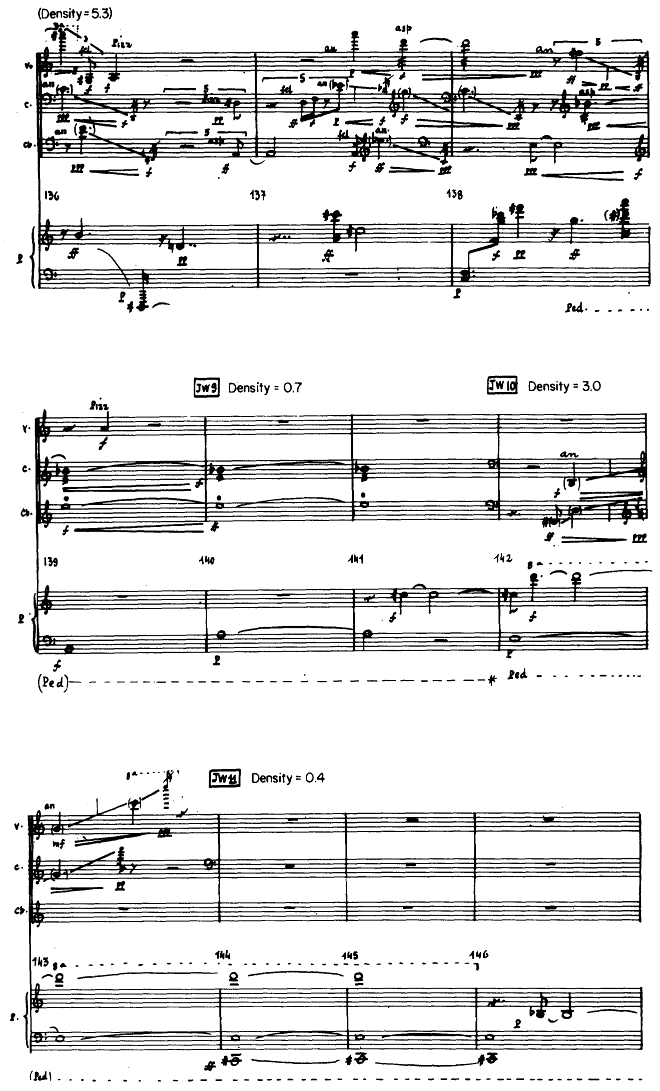

**Fig. 9. Iannis Xenakis, *Morsima-Amorsima*, 1962, measures 136–146. Each 'JW' mark signifies the first complete measure of a section; densities calculated by the program are presented in their 'objective' forms. The abbreviations in the string parts are: *asp*—arco sul ponticello; *an* —arco position normale, and *fcl*—frappe col legno. Copyright 1967 Boosey and Hawkes Music Publishers Limited, 295 Regent Street, London W1R 8JH, United Kingdom. Reprinted by permission of the publisher.**

eight such classes. The relative distributions in which the stochastic music program employs these timbre classes are graphed along the range of subjective densities in Fig. 8; an excerpt from *Morsima-Amorsima* appears in Fig. 9.

Once provided with its basic input data—length of the composition, average length of a section, range of densities and constitution of the ensemble—the stochastic music program is ready to go. The program is structured in two loops: an *outer* loop determines the duration, density and distribution of timbre classes (the last by locating the subjective density along the horizontal scale in Fig. 8, then determining weights for each timbre class along the vertical) for each section of the work, and an *inner* loop composes the actual notes. Details of these mechanisms may be found in Xenakis's own description of the program [45] and John Myhill's less mathematically demanding critique [46].

### Gottfried Michael Koenig

A less well-known but equally significant contributor has been Gottfried Michael Koenig, who has been associated for many years with the Institute of Sonology in Utrecht both as a teacher and as an administrator. Koenig's work in automated composition began in 1964 with a composing program entitled PROJECT1, and this program has been responsible for several of Koenig's compositions including his 1965 *Project 1, Version 1* for 14 instruments. Although Koenig's background is serial (he had worked early on as a composer, assistant and teacher in the Cologne electronic music studio), it had become apparent to Koenig by the time he undertook PROJECT1 that "the trouble taken by the composer with series and their permutations has been in vain; in the end it is the statistical distribution that determines the composition" [47].

A more elaborate program from 1969 called PROJECT2 [48] incorporates a broad palette of statistical procedures which users can patch together in different combinations. Although Koenig developed PROJECT2 for pedagogic functions, he has used it himself in one instance to create an ambitious *Übung für Klavier* ("Study for Piano", 1970). The score to this work consists of 12 "structures";

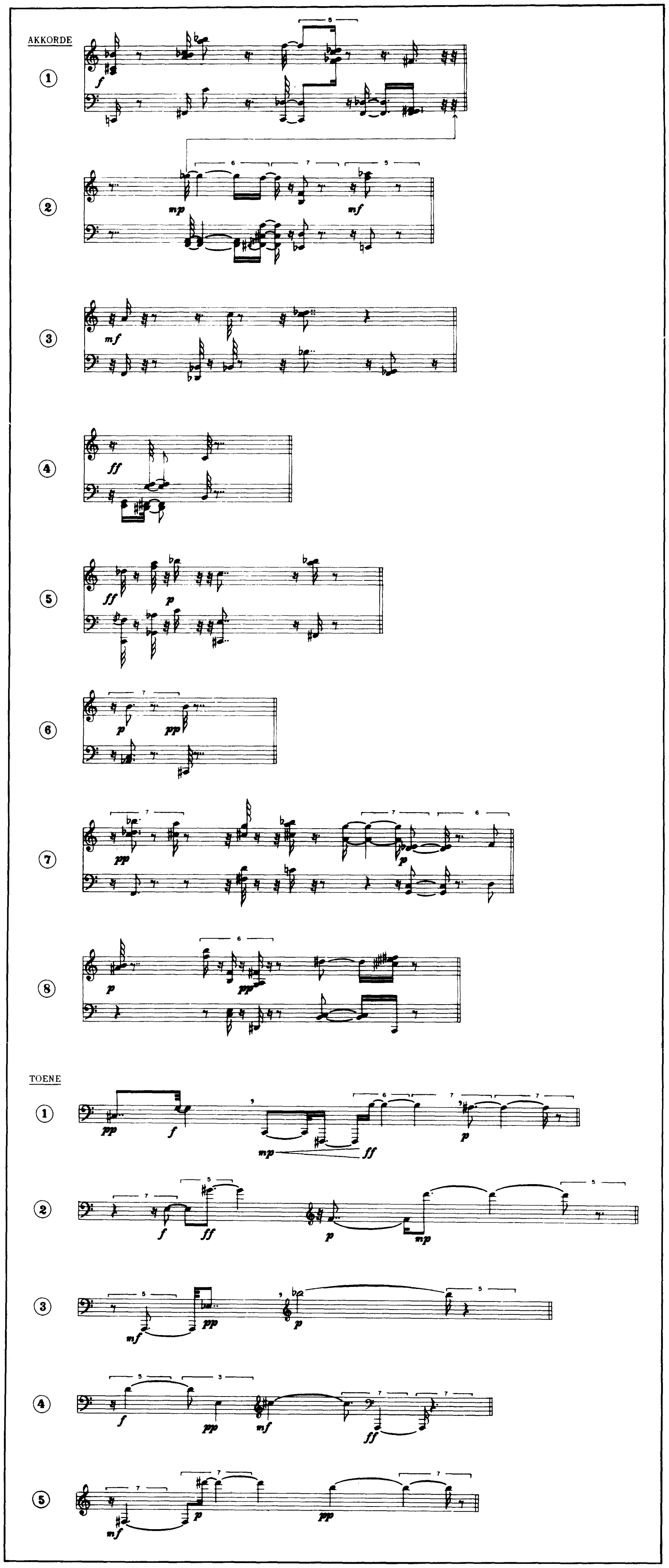

**Fig. 10. Gottfried Michael Koenig *Übung für Klavier*, 1970, Structure 8, Variant 1. The 'chords' (Akkorde) and the 'tones' (Toene) result from independent sets of directives to Koenig's PROJECT2 program. Copyright 1969 Gottfried Michael Koenig. Used by permission of the composer.**

each structure appears in three "variants", for which the only change in directives to PROJECT2 is the duration. Koenig acknowledges the indeterminate aspects of his compositional process by allowing the performer to choose between variants; the fact that such indeterminacy is closely circumscribed is reflected in Koenig's instructions that at least one variant of each structure *must* be played, and that all 12 structures must be presented in a prescribed order.

An examination of Structure 8 of the *Übung für Klavier* illustrates the workings of PROJECT2. Variant 1, which is the shortest, contains two sets of material: eight 'groups of chords' and five 'groups of tones' (Fig. 10). Koenig provides the following synopsis of Structure 8 in his instructions to the performer, which allow elements of performer discretion much like those controlling the performance as a whole:

> A transition: chords and tones alternate. There are also several groups of both chords and tones in every variant. Not all the groups of chords or tones have to be played, but those selected must be in the given order. Groups of tones must always be separated by chords, groups of chords can join on to one another (without jumping over groups of chords). A second group of chords then joins on rhythmically where indicated by the arrow. Such arrows, when present, have no significance if groups of chords alternate with groups of tones. What must be played are: at least three groups of tones in the first variant.... The number of groups of chords is left to the player, but he should begin with a group of chords [49].

Included among the musical "parameters" selected by PROJECT2 are (1) *harmony*, or degree of the chromatic scale, (2) *register*, (3) *entry delay*, or period between consecutive attacks, (4) *duration* and (5) *dynamic*. The *harmony* parameter is chosen by one of three *principles* (implemented as program subroutines) called ROW, CHORD and INTERVAL. For each parameter other than *harmony*, the user either must specify a constant value or must provide a *supply* of options and specify one of six *selection features* (also implemented as subroutines): SEQUENCE, ALEA, SERIES, RATIO, GROUP or TENDENCY. Table 1 presents a breakdown of the principles, selection features and supplies used to compose Structure 8 [50].

Of the *selection features* employed for Structure 8, ALEA and SERIES are the most fundamental. ALEA samples elements at random *with replacement* from the supply, with uniform likelihood of selection. SERIES also samples elements at random, but *without replacement*; i.e. each time SERIES selects an element, it eliminates this element from consideration until the entire supply has been exhausted (at this point, the full supply once again becomes available). The procedure for SERIES is reminiscent of the random shuffling undertaken by Brün's *Sonoriferous Loops* program. Compared to ALEA, SERIES requires many fewer calls before its actual choices come to reflect the distribution of options inherent in the supply. GROUP is a higher-order selection feature which repeats each element a number of times before resampling the supply; the number of repetitions and the new supply element can be selected either by ALEA or by SERIES, depending on the user's directives.

For the most part, the selection processes undertaken by PROJECT2 are independent of one another; e.g. pitches do not directly affect rhythms. However, in a few situations such as the restriction upon durations in Structure 8's "groups of chords", the user is able to engage a "block", causing PROJECT2 to toss out unacceptable choices and select new options.

**Table 1. Breakdown of principles, selection features and supplies used to compose Structure 8 of the *Übung für Klavier* (see Fig. 10)**

| Parameter | Chords | Tones |
|---|---|---|
| Harmony | CHORD from chord table shown in Figure 11. | INTERVAL from intervallic matrix shown in Figure 12. |
| Register | Constant: C3 — B6 | Constant: G2 to F#6 |
| Entry Delay | ALEA from 0.24, 0.30, 0.37, 0.46, 0.58, 0.72, and 0.89 seconds for entire chord. | ALEA from 0.46, 0.58, 0.72, 0.89, 1.11, 1.38, and 1.72 seconds. |
| Duration | ALEA from 0.10, 0.12, 0.15, 0.19, 0.24, 0.30, 0.46, 0.58, and 0.72 seconds for entire chord; duration may not exceed entry delay. | Same as entry delay |
| Dynamics | GROUP: elements by SERIES from pp, p, mp, mf, f, and ff; repetitions by SERIES from 3, 4, 5, 6, and 7. | SERIES from pp, p, mp, mf, f, and ff. |

Like GROUP, Koenig's 'harmonic' *principles*, CHORD and INTERVAL, may be regarded as 'higher-order' selection features. Each call to CHORD first requests a module such as ALEA or SERIES to choose an entry from a table of chords provided by the user. Koenig's table for the "groups of chords" in Structure 8 is shown here as Fig. 11; in this instance, the job of selection falls to ALEA. Once it has obtained a chord, CHORD again invokes either ALEA or SERIES to determine a (register-free) transposition; ALEA also received this second job in Structure 8.

The INTERVAL principle employed for the 'groups of tones' implements a Markov chain: In order to select a *current* interval, INTERVAL begins by consulting a user-provided logical matrix, such as the one for Structure 8 shown in Fig. 12. This matrix informs INTERVAL which of the 11 intervals from the minor second to the major seventh are acceptable, given the *most recent* interval employed by the program. INTERVAL then requests SERIES to select from these acceptable options. The *current* interval then becomes the *most recent* interval for the next call to INTERVAL, perpetuating the chain.

Once it has determined a chromatic degree using one of Koenig's *principles*, it remains for PROJECT2 to generate lower and upper registral limits for this degree. In Structure 8, Koenig has specified constant registral limits. Thus, each pitch within the "groups of chords" will be placed randomly within any of the four octaves between C3 (C below middle C) and B6. Similarly, pitches in the "groups of tones" occur randomly between G2 and F#6.

| Index | Degrees | | | | | |
|---|---|---|---|---|---|---|
| 1 | C# | F | | | | |
| 2 | | | F# | A | | |
| 3 | | | | | Bb | C |
| 4 | | F | F# | | | |
| 5 | | | | A | Bb | |
| 6 | C# | F | F# | | | |
| 7 | C# | F | | | | C |
| 8 | | | F# | A | Bb | |
| 9 | | F | F# | A | | |
| 10 | C# | | | | Bb | C |
| 11 | | | | A | Bb | C |
| 12 | C# | F | F# | A | | |
| 13 | C# | F | | | Bb | C |
| 14 | | | F# | A | Bb | C |
| 15 | | F | F# | A | Bb | |
| 16 | C# | F | F# | | | C |
| 17 | C# | | | A | Bb | C |

**Fig. 11. Chordal table for Structure 8 of Koenig's *Übung für Klavier*.**

| Most Recent Interval | Current Interval | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | m2 | M2 | m3 | M3 | P4 | TT | P5 | m6 | M6 | m7 | M7 |
| m2 | yes | no | no | no | no | no | no | no | no | yes | yes |
| M2 | no | yes | yes | yes | no | no | yes | yes | no | yes | yes |
| m3 | no | yes | no | no | no | no | yes | no | yes | no | no |
| M3 | no | no | no | no | no | yes | no | yes | no | yes | no |
| P4 | no | no | no | no | no | no | yes | no | yes | yes | no |
| TT | no | yes | no | no | no | yes | no | yes | no | no | no |
| P5 | no | yes | yes | no | yes | no | no | no | no | no | no |
| m6 | no | yes | no | yes | no | no | no | no | no | yes | no |
| M6 | no | no | yes | no | yes | no | no | no | no | yes | no |
| m7 | yes | yes | no | yes | yes | yes | no | no | yes | yes | no |
| M7 | yes | yes | no | no | no | no | no | no | no | no | yes |

**Fig. 12. Intervallic matrix for Structure 8 of Koenig's *Übung für Klavier.***

### Otto Laske

A close collaborator of Koenig's has been Otto Laske [51–57]. As a composer and musical epistemologist, Laske has been strongly interested in the impact of computer-aided tools on the way composers think. For a number of years beginning in the early 1970s, Laske has worked to blend insights from cognitive psychology, formal linguistics and artificial intelligence into an *expert system* "bridging the gap between a composer's semantic intuitions and low-level data". In Laske's work, composing programs such as Xenakis's and Koenig's are often employed to create "proto-scores" which Laske will freely edit and arrange.

## V. INTERACTIVE UTILITIES IN THE UNITED STATES AND CANADA: 1970–1978

The 1970s marked the widespread obsolescence of 'batch-oriented' computers programmed by stacks of punched cards in favor of 'on-line' systems, either in the form of single-user computers and minicomputers dedicated to real-time functions or of large time-shared systems capable of servicing many users simultaneously. The new fashion for on-line computing found its musical counterpart in utilities motivated largely by a desire to facilitate rapid interaction between composer and sound: Max Mathews and Richard Moore's 1968 GROOVE [58–63]; Leland Smith's 1972 SCORE [64] (later converted into a music printing program); Barry Truax's 1973 POD with its subsequent enhancements [65–68]; Herbert Brün's 1976 SAWDUST [69] and William Buxton's 1978 SCED [70–73]. Such utilities were coupled inevitably with sound synthesis, at first through digitally controlled analog synthesis or through software sound-synthesis packages modeled after Mathews's MUSIC4 and MUSIC5 programs [74], and later through digital hardware.

### Ghent and Spiegel

In particular, an environment such as Mathews and Moore's GROOVE (Generating Realtime Operations on Voltage-controlled Equipment) [75] utilizes the computer as an intermediary between the user (working through specialized control devices such as keyboards and/or panels of knobs and switches, along with the usual alphanumeric terminal) and an array of real-time sound-synthesis hardware. Such an environment enables a composer/performer to deal at a reflex level with sounds and timings, although the price for this is that compositional procedures must be simple enough to run in real time. Along with random selection, some of the more familiar procedures falling within this category are traditional motivic operations. The most active users of GROOVE were Emmanuel Ghent beginning in 1968 [76–78] and Laurie Spiegel beginning in 1974 [79–81].

The first piece to involve the GROOVE facility directly in compositional decision making was Ghent's 1974 *Lustrum. Lustrum* is scored for a quintet of special electronic stringed instruments developed by Mathews (among other things, these instruments could be set to emulate brassy timbres), for conventional brass quintet and for computer-generated tape; the piece also exists in an all-computer version entitled *Brazen.*

*Lustrum* (or *Brazen*) has its genesis in the 15-element "pitch-set" depicted in Fig. 13 and in an analogous set of durations. In order best to "hear" what the computer would be doing in real time as he manipulated GROOVE's input devices during this initial foray into automated composition, Ghent employed a pitch set with strong tonal orientation around B$^b$. When he had specified both pitch and duration sets, he developed a program to generate 10 "functions" (monodic sequences of notes); these functions were produced by using weighted randomness to select pitch-set elements and duration-set elements for each note.

The resulting functions, denoted by Ghent using the symbols F1, F2, ... F10, in turn provided all of the material for *Lustrum.* Because Ghent's programs represented pitches in these functions by their position in the set rather than by their location on the keyboard, he was able to implement motivic operations which acted *indirectly* on set positions—operations not unlike the traditional practices of transforming motifs *modally*—relative to notes of a constituent major or minor scale. In Fig. 14, for example, violin II plays the original form of F1. The 'cello plays a derivative of F1 which is obtained by inverting this original sequence around 8, the central position in Fig. 13: the opening D in violin II lies two positions *lower* than position 8, so the opening G in the 'cello lies two positions *higher*, and so forth. Also appearing in Fig. 14 is F2, played in original form by the viola, in 'indirect' inversion by trumpet 2, in retrograde by trumpet 1 and in retrograde inversion by the bass. Not shown in this excerpt is the operation of indirect transposition, which Ghent calls *translocation.* Translocation is performed by adding a constant *modulo 15* to each successive position in a function.

Concerning the role of the computer in this process, Ghent states:

The use of the indirect pitch referencing technique ... is indeed possible without the help of a computer but the labor involved in spelling out all manner of possible variants would be prohibitive. Also the more one hears what the computer is producing in response to one's own selective processes, the more ideas suggest themselves as further avenues for variation [82].

Subsequent to *Lustrum*, Ghent developed compositional algorithms based on the "linearization" of chordal structures [83]. Since he had already employed similar techniques in manually composed works for small ensembles, Ghent regarded his programming as "teaching the computer to think the way [Ghent] thought, compositionally"—although Ghent re-

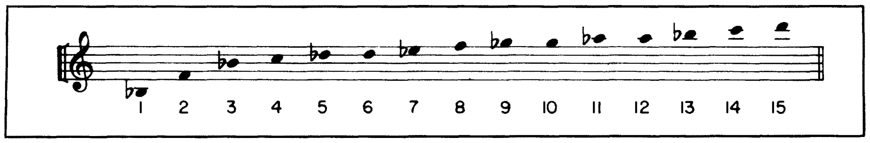

**Fig. 13. Pitch set for Emmanuel Ghent's *Lustrum.* Redrawn from Ghent, "Real-Time Interactive Compositional Procedures" [58].**

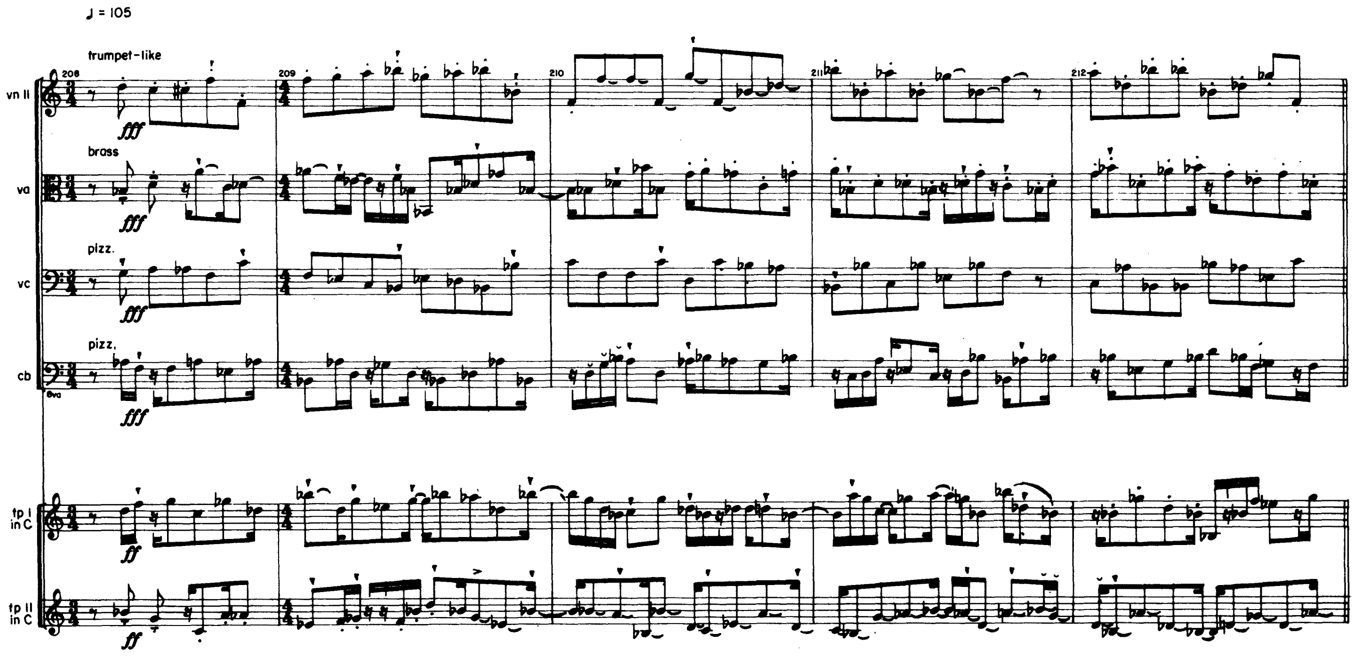

Fig. 14. Emmanuel Ghent, *Lustrum*, 1974, measures 208–212. Copyright 1978 Persimmon Press, 131 Prince Street, New York, NY 10012, U.S.A. Used by permission of the publisher.

served the option to modify what the computer produced.

A variety of comments by Spiegel illustrates how a composer might typically interact in real time with GROOVE. To create her 1974 composition *Patchwork*, for example, Spiegel developed programs that monitored GROOVE's input devices in order to "derive from [them] much more complex music than [Spiegel] actually played". Spiegel has commented as follows:

The program I wrote for [*Patchwork*] had all Bach's favorite [motivic] manipulations—retrograde, inversion, augmentation, diminution, transposition—available on switches, knobs, pushbuttons, and keys, so that I could manipulate the 4 simple melodic and 4 rhythmic patterns with them in the same way that a player of an instrument manipulates individual tones. (I did edit it a lot, too.) [84]

The melodic patterns mentioned by Spiegel were represented by her programs as "relative (intervallic) distances" from a "pitch reference point" which Spiegel could control in real time from her keyboard. Elsewhere Spiegel describes some of the 'higher-level' methods of controlling the musical evolution of her 1975 *Music for Dance* through GROOVE's input devices:

Because I could hear the music played while computation was in progress, I could choose which musical decisions I wanted to make, leaving the rest to follow the logic I had coded.... For example, I could set the probability that a certain pitch would be played on strong beats at zero percent, at 100 percent, or anywhere in between, by turning a knob while listening to the music evolve [the position of the knob at each moment would be recorded by GROOVE for later playback]; or I could throw a switch to change over to an entirely different set of rules [85].

### Truax and Buxton

Two Canadians who had studied under Koenig and Laske, Barry Truax and William Buxton, have since organized computer music studios respectively in Vancouver and Toronto. In addition to developing real-time performance systems, both Truax and Buxton have developed non-real-time environments which permit users to get by with little or no programming. These environments enable users to monitor results closely as a composition takes shape. They minimize delays in non-real-time interaction by providing a menu of precompiled "macros" (Buxton's term) and by incorporating facilities for rapidly evaluating musical products, specifically: real-time digital synthesis augmented visually by graphic displays. Laske [86] has documented how he built up his 1980 composition *Terpsichore* as a succession of increasingly elaborate "subscores", using macros supplied by Buxton's SCED (SCore EDiting) utility.

## VI. NEW APPROACHES: 1976–1986

Recent efforts have shown a new sophistication toward automated decision making along with increased machine participation in decisions affecting large-scale compositional form. Included among the 'new generation' of composers seeking creative assistance from computers are Larry Austin, Claudio Baffioni [87], Clarence Barlow [88–90], Tommaso Bolognesi [91], Lelio Camilleri, Joel Chadabe, Thomas DeLio, Charles Dodge [92], David Feldman, Christopher Fry [93], Peter Gena [94], Francesco Guerra [95], Christos Hatzis [96], Steven Holtzman [97, 98], Kevin Jones [99], Gary Kendall [100], Shirish Korde, Petr Kotik, Ron Kuivila, Laura Tedeschini Lalli [101], David Levitt [102], Denis Lorrain [103], Leonard Manzara [104], Larry Polansky, Curtis Roads [105–108], David Rothenberg [109], Andrew Schloss, William Schottstaedt [110], Sever Tipei [111], Horacio Vaggione, Charles Wourinen, David Zicarelli and myself [112–120]. The extent of this new activity has been far too great to continue with a detailed survey of pieces, so I shall limit my remaining examples to significant works that have not been described previously in the literature. Further details may be obtained from the works cited in the References and Notes.

### Thomas DeLio

Thomas DeLio's 1976 *Serenade* for solo piano is an elaborate work for which the form was composed manually while the details were selected automatically. The *Serenade* has three parts, each consisting of 10 sections; marked contrasts and occasional long silences between many of these sections give the *Serenade* an episodic quality congenial to its title. The work is built up from a number of primary rhythmic patterns and chromatic cells. Part I introduces these ideas in "embryonic states"; they emerge "fully formed" in Part II, which also serves a developmental purpose; Part III extrapolates several of the ideas from earlier parts into "broad sweeping guestures".

Figure 15 reproduces Part I, Section 10 of the *Serenade*. The basic ideas employed in this section are a 7:4 polyrhythm and two chromatic cells: G# B C D and its reflection, G# A# B D. Registers are distributed uniformly over the range from middle C upward. The section consists of two gradual evolutions, each lasting 17 quarters. The following breakdown of methods for handling rhythm and pitch is based upon a letter from DeLio to the author:

*Rhythm*. Over the first 17 quarters, the rhythm coalesces from a sparse, irregular texture to a consistent 7:4 pattern. DeLio accomplished this evolution by treating each quarter as a statistical *frame* whose elements are the 29 unequal rhythmic units obtained by interfering septuplet thirty-seconds against duple sixty-fourths. A graphic depiction of the resulting pattern appears in Fig. 16. In this section of the *Serenade*, only one note may attack at a time; the number of attacks per quarter results from a "first-order transition process" (i.e. a Markov chain) in which the number of attacks may either stay the same, increase by one or (with relatively small probability) decrease by one from one quarter to the next. This process resulted in the following chain:

5 6 6 7 7 7 8 7 8 8 9 9 10 10 10 10 10 10 ...

Once these numbers have been determined, DeLio's program locates these attacks within each quarter by selecting *without replacement* from the units depicted in Fig. 16. Initially, each unit receives equal weight; as the rhythm evolves, the off-beat thirty-seconds and sixty-fourths receive less and less weight until only the 7:4 pattern remains. This pattern holds consistently through the remaining 17 quarters.

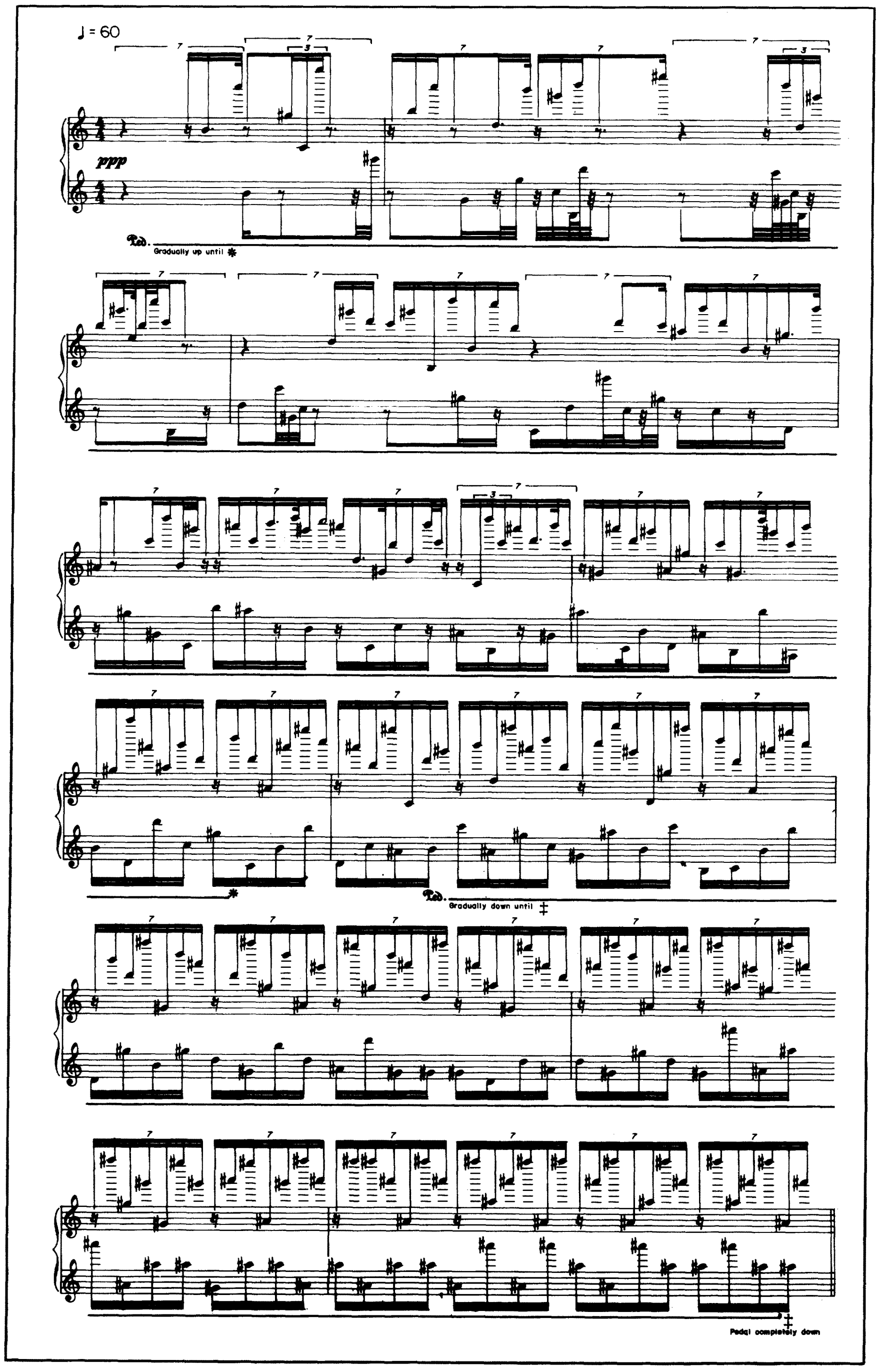

Fig. 15. Thomas DeLio, *Serenade*, 1976, Part I, Section 10. Copyright 1976 Thomas DeLio. Reprinted by permission of the composer.

*Pitch*. At the beginning of Section 10, DeLio's program selects degrees uniformly *with replacement* from G#, B, C and D. Through the first 17 quarters the likelihood of selecting C decreases gradually while the likelihood of selecting A# gradually increases, so that G#, A#, B and D receive uniform weight at the midpoint of the section. Over the remaining 17 quarters, the weights for G#, B and D decrease gradually until only A# remains at the end of the section.

## Grammars

The effectiveness of composing programs has been improved radically through the introduction in recent years of *recursive* programming techniques; the first major area of activity relying heavily on such techniques has been the application of Noam Chomsky's *phrase-structure grammars* to composition. This approach enables a composer to describe musical forms economically, by first providing a general *archetype* (or *axiom*) of the form and by further listing a set of *productions* for deriving details from generalities. The full power of such an approach can be attained only if the productions are capable of acting upon their own results (i.e. capable of *recursion*).

Although Curtis Roads had already described an elaborate *left-to-right* composing program called PROCESS/ING as early as 1975 [121, 122], his advocacy of *top-down* Chomskian methods [123] has achieved the most influence. Roads's emphasis upon productions classified by Chromsky as *context-free* was quickly extended to embrace other Chomskian ideas such as *context-sensitive* productions and simple *transformations* in Steven Holtzman's 1980 "A Generative Grammar Definitional Language for Music" [124]. In Kevin Jones's 1981 "stochastic web grammars" [125], context-free productions are employed to carve up a region of musical space (described in

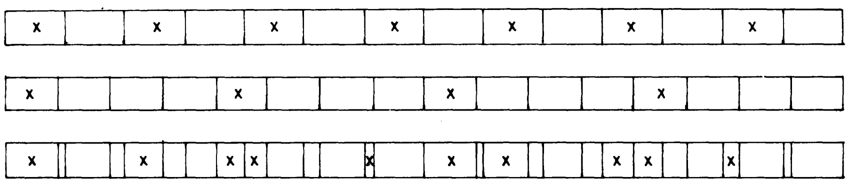

Fig. 16. Interferences resulting from a 7:4 polyrhythm. X's indicate 'primary' units.

Jones's simplest examples by boundaries for time and pitch coordinates) into progressively smaller chunks until specific descriptions of notes have been obtained. None of the speculations by Roads, Holtzman or Jones resulted, to my knowledge, in more than a few brief, illustrative musical examples. Ironically, my own 1980 *Crystals* for string orchestra [126] resulted from top-down productions similar to Jones's. However, my methods were derived in response to the Gestalt hierarchies of James Tenney—only afterward did I discover what Roads, Holtzman and Jones had done.

Tenney himself has resumed involvement with composing programs after a 20-year hiatus. In his 1983/1984 *Bridge* for two retuned pianos, eight hands, Tenney fuses hierarchic ideas from his Bell Labs period with a slightly more recent interest in the harmonic possibilities of just intonation. *Bridge* consists of three sections lasting approximately 8, 13 and 21 minutes. The initial section presents a wholly random environment evocative of the compositional world of John Cage; the final section realizes a Gestalt hierarchy in which individual notes combine into "elements" (chords), elements combine into "clangs", clangs combine into "sequences", which ultimately combine into the section as a whole; the middle section effects a "bridge" from Cage's world to Tenney's. An excerpt from the final section appears in Fig. 17. The notated pitches are modified by the tuning system shown in Fig. 18.

A third independent path to recursive compositional procedures has been *fractal geometry*. Although popularly associated with Benoit Mandelbrot, fractals are grounded, in fact, in recursive notions which were well understood by turn-of-the-century mathematicians such as Georg Cantor (inventor of the "Cantor Set") and Giuseppe Peano (inventor of the "Peano Curve"). Mandelbrot's writings proved a source of inspiration to Charles Wourinen, who has explored fractals both in a set of short computer-composed studies realized around 1980 and in more recent manually composed works. Fractals have since provided the impetus for several computer-composed pieces including Larry Austin's 1981 *Canadian Coastlines*, Horacio Vaggione's 1984 *Fractal C* and Charles Dodge's 1984 *Profile* [127].

## Artificial Intelligence

An especially potent approach that also relies on recursive programming techniques has drawn from Artificial Intelligence in order to implement decision-making processes that employ *searches* to discriminate actively between multiple options. Such searches typically incorporate one or more *prioritizations*—functions that measure in numeric terms (positive or negative, depending on the application) how each option available to a decision will contribute to the resulting musical context. In the author's terminology [128], search-based programs divide into two classes:

1. *Comparative* searches systematically enumerate *every possible* configuration of decisions, evaluating each configuration in order to determine the 'best' possible solution to a problem.
2. *Constrained* searches improve dramatically upon the random-sampling-and-testing approach of Hiller and Isaacson's *Illiac Suite*. Each time such a program initiates a decision, a constrained search *prioritizes* all the available options from most to least desirable and organizes these options into a *schedule*. It then methodically works through this schedule until it finds an option that passes all the constraints. Should every option in a schedule prove unacceptable, the program then *backtracks*, revises one or more earlier decisions and tries again.

Although computer scientists have employed recursive searches in chess-playing and theorem-proving programs since the mid-1950s—and despite Stanley Gill's demonstration with his 1963 computer-composed *Variations on a Theme by Berg* [129] that backtracking could be an effective tool in composing programs—the approach has mostly been ignored until very recently. This pattern was broken by Larry Polansky's 1975 *Four-Voice Canons*, composed by a program with non-rigorous backtracking, and independently by Kemal Ebcioğlu's 1980 modeling [130] of compositional process in two-part sixteenth-century counterpoint. Ebcioğlu's was the first program since Gill's to implement a fully rigorous constrained search with scheduling and backtracking. He has since employed such methods with great success in simulating J.S. Bach's procedures for harmonizing chorales [131]; similar success in chorale harmonization has been documented by Marilyn Thomas [132].

I employed comparative searches to create my own 1981 *Protocol* for piano [133]; but since that time I have created a number of original compositions using constrained searches [134]. In particular, my method of 'statistical feedback' [135] has provided a determinate strategy for reconciling long-term statistical distributions with short-term stylistic and idiomatic constraints. The set of composing programs I developed for the U.S. pavilion at Expo '85 in Tsukuba, Japan [136], and the programs for my 1986 solo violin piece, *Concurrence* [137], employ 'linked' data structures, such as 'lists', 'trees' and 'networks', to represent the complex interrelationships between elements both of the musical scores themselves and of the musical 'knowledge base' consulted by the programs as they fabricate these scores.

Many of the more recent developments in automated composition would probably not have been possible without supporting advances in computer technology. Backtracking has been known for decades, but it simply was not practical without either liberal grants of computer time or dedicated systems that could be left running for many hours (often overnight, in my own experience). The newest home computers have changed all that. A second important factor has been the dramatic rise in ceilings on both 'random-access' and 'mass-storage' memory, coupled with the introduction of 'virtual' operating systems which allow a program to utilize more random-access memory than might physically be present in the computer. Where the original ILLIAC employed by Hiller and Isaacson was limited to 1024 words of random-access memory for both composing program and data, modern home computers can access *megabytes* of information rapidly. Only the barest fraction of this potential has been exploited to date.

## Interactive Utilities of the 1980s

The tradition of 'interactive' compositional utilities has been extended into the 1980s with work undertaken by Curtis Roads and David Levitt at the Massachusetts Institute of Technology.

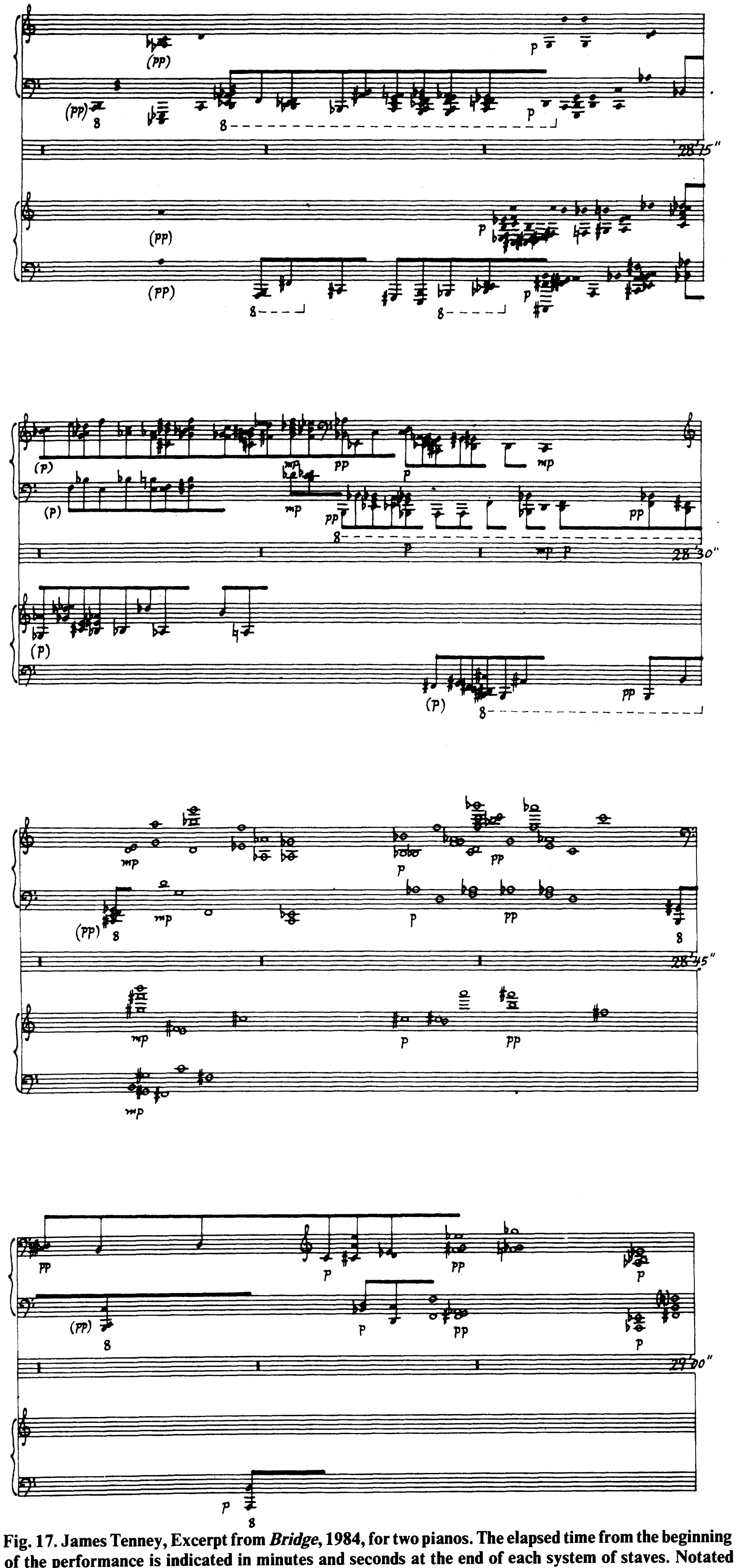

**Fig. 17. James Tenney, Excerpt from *Bridge*, 1984, for two pianos. The elapsed time from the beginning of the performance is indicated in minutes and seconds at the end of each system of staves. Notated pitches are modified by the tuning system shown in Fig. 18. The rhythmic notation shows relative attack-points by their horizontal placement on the page. Unbeamed, solid noteheads indicate staccato; beamed notes last either up to the next stem or to the end of the beam; open noteheads employ pedal. Copyright 1985 Sonic Art Editions, 2617 Gwynndale Avenue, Baltimore, MD 21207, U.S.A. Used by permission of the publisher.**

Roads's 1982 IOS [138] is a utility for "interactive orchestration based on score analysis", in which parsing and pattern-matching algorithms are employed to isolate musical "objects" for user modification. (Although Roads has cited IOS as a musical application of Artificial Intelligence, his descriptions of this program indicate that IOS makes few decisions on its own and that the program makes no use whatsoever of recursive searches.) David Levitt's 1986 KIT [139] is a graphic editor that enables users to patch together icons representing elementary operations on streams of note data into real-time networks. Included among the most elementary of Levitt's operations might be input from a keyboard, button or slider (physical or virtual), output to a synthesizer, generators such as clocks or units analogous to the "selection features" of Koenig's PROJECT2 and modifiers such as adders and delay lines for loops. Users may define their own higher-order icons by patching together simpler units.

## VII. CONCLUSIONS

For most of the composers discussed in this article, composition has *not* been a vehicle of 'expression' or 'affect' in the usual senses of these words. Rather, these composers expect listeners to judge their musical "artifacts", as Barbaud called them, on the basis of more abstract aesthetic qualities such as balance, clarity, depth (in the hierarchic sense) and so forth. They have used computers to create these artifacts because they believe that such qualities are quantifiable—at least within the scope of a particular work—and they have concluded from experience that there are many compositional problems that computers are better equipped than humans to solve.

Considered individually, many of the earlier efforts described in this article have been quite simple. This simplicity has been due partially to limits imposed by technology and partially to the spirit of the 1960s and 1970s, when economy of musical process was itself considered a virtue. Today, a one-pass, self-contained composing program such as Ebcioğlu's Bach simulator [140] might involve as many as 7,000 lines of highly concise code. An alternative approach employed by DeLio, Barlow [141] and myself [142] has been to divide the overall compositional process into multiple phases, each undertaken by a separate program.

Taken collectively, the compositions surveyed here present a rich diversity of techniques for programming computers to compose music. Practically all the

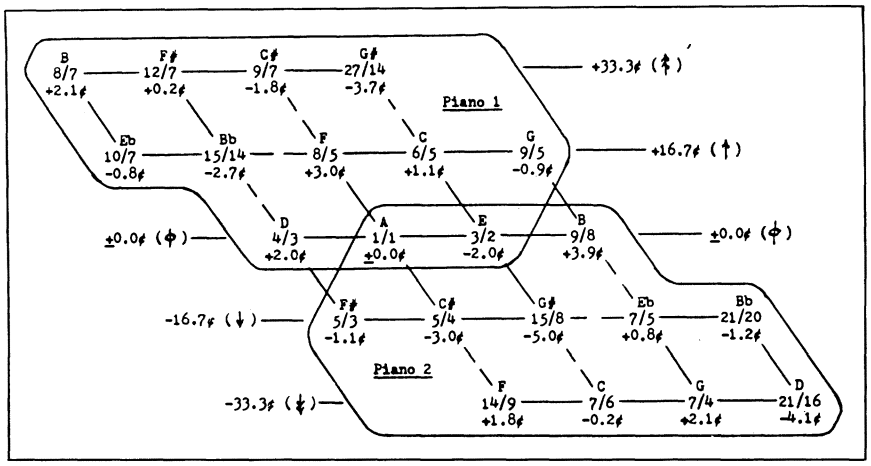

**Fig. 18. Tuning system for James Tenney's *Bridge*.**

approaches employed today have precedents in the groundbreaking years from 1956 to 1976: constrained decision making (Klein and Bolitho, Hiller and Isaacson), conditional or Markov randomness (Hiller and Isaacson, Hiller and Baker, Koenig's INTERVAL feature), statistical procedures (Xenakis, Brün, Koenig), evolutions (Tenney, Myhill, Koenig's TENDENCY feature [143], DeLio), hierarchic, (grammatic) structures (Tenney), motivic transformations Barbaud, Hiller and Baker, Ghent, Spiegel), backtracking (Gill). Although a number of the early computer-composed pieces might seem coarse in comparison to direct human efforts, the more recent introduction of recursion, linked data structures and prioritized decision making now enable composing programs to be 'fine tuned' until their results equal or surpass what humans can achieve by hand.

With the musical industry becoming increasingly 'high-tech', and with today's teenager acquiring programming expertise once reserved for veterans of university graduate programs, the notion of a composer/programmer is no longer anathema. Growing numbers of composers are coming to regard computers not only as *legitimate* creative tools but as *effective* tools at that. Although how deeply the computer will penetrate into such traditionally 'human' domains is yet to be gauged, it is clear already that automated composition is an established reality of our time.

**Acknowledgements**—Special acknowledgement is due to Lejaren Hiller for allowing me free access to his extensive files on early computer composition and to the late John Myhill for critiquing this manuscript. I would also like to thank the following people for documents, scores and analytic insights: Herbert Brün, William Buxton, Thomas DeLio, Emmanuel Ghent, Gottfried Michael Koenig, Otto Laske, Larry Polansky, Curtis Roads, Laurie Spiegel, James Tenney, Barry Truax and Iannis Xenakis.

## REFERENCES AND NOTES

1. L. Hiller, "Programming the I Ching Oracle", *Computer Studies in the Humanities and Verbal Behavior* 3, 130 (1970).
2. L. Hiller and C. Ames, "Automated Composition: An Installation at the 1985 International Exposition in Tsukuba, Japan", *Perspectives of New Music* **23**, No. 2, 196 (1985).
3. L. Hiller and R. Baker, "*Computer Cantata*: A Study of Compositional Method", *Perspectives of New Music* **3**, No. 1, 62 (1964).
4. L. Hiller and R. Kumra, "Composing *Algorithms II* by Means of Change Ringing", *Interface* **8**, No. 3, 129 (1979).
5. L. Hiller and L. Isaacson, *Experimental Music* (New York: McGraw-Hill, 1959). Reprinted 1979, Greenwood Press.
6. J. Myhill, "Some Simplifications and Improvements in the Stochastic Music Program", *Proceedings of the 1978 International Computer Music Conference* (Computer Music Association, 1978).
7. I. Xenakis, *Formalized Music: Thought and Mathematics in Composition* (Indiana University Press, 1971).
8. L. Hiller, "Music Composed with Computers—A Historical Survey", Chap. 4 in *The Computer and Music*, Harry Lincoln, ed. (Ithaca, NY: Cornell University Press, 1970).
9. Anon., "Syncopation by Automation", *Data from ElectroData*, Aug. 1956 (Pasadena, CA: Electro-Data Division of Burroughs, Inc.).
10. It was aired over KABC-TV, Los Angeles, CA.
11. Hiller and Isaacson [5].
12. Hiller and Baker [3].
13. J. Tenney, "Computer Music Experiences, 1961-1964", *Electronic Music Reports* **1**, No. 1, 23 (1969).
14. T. Blum, "Herbert Brün: Project SAWDUST" (review), *Computer Music Journal* 3, No. 1, 6 (1979).
15. H. Brün, "From Musical Ideas to Computers and Back", Chap. 2 in *The Computer and Music* [8].
16. H. Brün, *Uber Musik und zum Computer* (Karlsruhe: G. Braun, 1971).
17. H. Brün, "Herbert Brün Compositions", booklet accompanying the record album of the same name, non-sequitur records 1-3, Champaign, IL, 1983.
18. Hiller and Ames [2].
19. Myhill [6].
20. J. Myhill, "Controlled Indeterminacy: A First Step towards a Semi-Stochastic Music Language", *Computer Music Journal* 3, No. 3 (1979). Reprinted in *Foundations of Computer Music*, C. Roads and J. Strawn, eds. (Cambridge, MA: M.I.T. Press, 1985). An upgraded formula for 'controlled indeterminacy' appears in Hiller and Ames [2].
21. Hiller and Baker [3].
22. Gyorgy Ligeti, "Pierre Boulez: Entscheidung und Automatik in der Structure Ia", *Die Reihe IV: Junge Komponisten*, K. Stockhausen and H. Eimert, eds. (Vienna: University Edition, 1958) p. 33. Translated by Leo Black, under the title "Pierre Boulez: Decision and Automatism in Structure Ia", *Die Reihe IV: Young Composers* (Bryn Mawr, PA: Theodore Presser, 1960) p. 36.
23. Hiller and Isaacson [5].
24. Hiller and Baker [3].
25. For a discussion of Tenney's activities as the first in what came to be a series of composers in residence at Bell Labs, see Tenney [13].
26. M. Mathews, with L. Rosler, "Graphical Language for the Scores of Computer Generated Sounds", *Perspectives of New Music* **6**, No. 2, 92 (1968).
27. M. Mathews, with J. Miller, F.R. Moore, J.R. Pierce and J.C. Risset, *The Technology of Computer Music* (Cambridge, MA: M.I.T. Press, 1969).
28. M. Mathews and F.R. Moore, "GROOVE: A Program to Compose, Store, and Edit Functions of Time", *Communications of the Association for Computing Machinery* **13**, No. 12, 715 (1970).
29. See Myhill [6] and Xenakis [7].
30. These *Gravesaner Blätter* articles were later collected into the 1971 book *Formalized Music*; see Xenakis [7].
31. Brün [15].
32. Xenakis [7].
33. Brün [17].
34. J. Schillinger, *The Schillinger System of Musical Composition*, 2 volumes (New York: Carl Fischer, 1941).
35. P. Barbaud, *Initiation à la composition musicale automatique* (Paris: Dunod, 1966).
36. P. Barbaud, "Algorithmic Music", from a trade publication (Paris: Systems Bull, 1969).
37. See Myhill [6] and Xenakis [7].
38. G.M. Koenig, Performance notes and "Analytical Appendix" to the *Übung für Klavier* (Utrecht: Institute of Sonology, 1969).
39. G.M. Koenig, "Project 1", *Electronic Music Reports* **1**, No. 2, 32 (1970).
40. G.M. Koenig, "PROJECT2: A Programme for Musical Composition", *Electronic Music Reports* **1**, No. 3 (1970).
41. G.M. Koenig, "Protocol", *Sonological Reports* **4** (Utrecht: Institute of Sonology, 1979).
42. O. Laske, "Composition Theory in Koenig's Project One and Project Two", *Computer Music Journal* **5**, No. 4, 54 (1981).
43. Xenakis [7].
44. See Myhill [6] and Xenakis [7].
45. Xenakis [7].
46. Myhill [6].
47. Koenig [39].
48. Koenig [40].
49. Koenig [38], "Analytical Appendix", p. 25.

50. Koenig [38].
51. O. Laske, "Towards a Musical Intelligence System: OBSERVER", *Numus-West* No. 3, 11 (1973).
52. O. Laske, "The Information-Processing Approach to Musical Cognition", *Interface* **3**, No. 2, 109 (1974).
53. O. Laske, "Toward a Theory of Musical Cognition", *Interface* **4**, No. 2, 147 (1975).
54. O. Laske, *Music, Memory, and Mind: Explorations in Cognitive Musicology* (University Microfilms, 1977).
55. O. Laske, "Subscore Manipulation as a Tool for Compositional and Sonic Design", *Proceedings of the 1980 International Computer Music Conference* (Computer Music Association, 1980).
56. O. Laske, *Music and Mind: An Artificial Intelligence Perspective* (Needham, MA: self-publication, 1980).
57. Laske [42].
58. E. Ghent, "Real-Time Interactive Compositional Procedures", *Proceedings of the American Society of University Composers* **9/10** (1975-1976).
59. E. Ghent, "Interactive Compositional Algorithms", *Proceedings of the 1977 International Computer Music Conference* (Computer Music Association, 1977).
60. E. Ghent, "Further Studies in Compositional Algorithms", *Proceedings of the 1978 International Computer Music Conference* (Computer Music Association, 1978).
61. Mathews and Moore [28].
62. L. Spiegel, Liner notes to *The Expanding Universe* (Philo Records, no. 9003, 1980).
63. L. Spiegel, Program Notes for *Music for Dance*, New York Philharmonic *Horizons '84* concert, 3 June 1984.
64. L. Smith, "SCORE: A Musician's Approach to Computer Music", *Journal of the Audio Engineering Society* **20**, No. 1, 7 (1972: reprinted 1973 in *Numus-West* No. 4); *SCORE Manual*, New version (Computer printout, Stanford University, Palo Alto, CA, 1974).
65. W. Buxton, *Manual for the POD Systems* (Utrecht: Institute of Sonology, 1975).
66. B. Truax, "Computer Music Composition: The Polyphonic POD System", *IEEE Computer* (Aug. 1978).
67. B. Truax, "The Inverse Relation between Generality and Strength in Computer Music Programs", *Interface* **9**, No. 2, 49 (1979).
68. B. Truax, "The PODX System: Interactive Compositional Software for the DMX-1000", *Computer Music Journal* **11**, No. 1, 29 (1985).
69. Blum [14].
70. W. Buxton, *Design Issues in the Foundation of a Computer-Based Tool for Music Composition* (Technical report CSRG-97, Computer Systems Research Group, University of Toronto, 1978).
71. W. Buxton, R. Sniderman, W. Reeves, S. Patel and R. Baecker, "The Evolution of the SSSP Score Editing Tools", *Computer Music Journal* **3**, No. 4, 14 (1979).
72. W. Buxton, *Music Software User's Manual* (Computer Systems Research Group, University of Toronto, 1980).
73. Laske [55].
74. Mathews et al. [27].
75. Mathews and Moore [28]. GROOVE represents yet another computer music innovation from Bell Labs.
76. Ghent [58].
77. Ghent [59].
78. Ghent [60].
79. Spiegel [62].
80. L. Spiegel, "Sonic Set Theory: A Tonal Music Theory for Computers", *Proceedings of the Second Annual Symposium on Small Computers and the Arts* (1982).
81. Spiegel [63].
82. Ghent [58].
83. Ghent [60].
84. Spiegel [62].
85. Spiegel [63].
86. Laske [55].
87. Claudio Baffioni, Francesco Guerra and Laura Tedeschini Lalli, "The Theory of Stochastic Processes and Dynamical Systems as a Basis for Models of Musical Structures", *Musical Grammars and Computer Analysis*, M. Baroni and L. Callegari, eds. (Florence: Olschki, 1984).
88. C. Abbot, "Clarence Barlow: Çoğluotobüsişletmesi" (review), *Computer Music Journal* **7**, No. 4, 66 (1983).
89. C. Barlow, "Bus Journey to Parmetron", *Feedback Papers*, Nos. 21-23 (1980).
90. Stephen Kaske, "A Conversation with Clarence Barlow", *Computer Music Journal* **9**, No. 1, 19 (1985).
91. T. Bolognesi, "Automatic Composition: Experiments with Self-Similar Music", *Computer Music Journal* **7**, No. 1, 25 (1983).
92. Charles Dodge, "*Profile*, A Musical Fractal", *Proceedings of the 1986 Symposium on the Arts and Technology at Connecticut College* (Shoestring Press, in press).
93. C. Fry, "Computer Improvisation", *Computer Music Journal* **4**, No. 3, 48 (1980); "Flavors Band: A Language for Specifying Musical Style", *Computer Music Journal* **8**, No. 4, 20 (1984).
94. P. Gena, *MUSICOL Manual, Version 1* (Technical report no. 7, Experimental Music Studio, State University of New York at Buffalo, 1973).
95. Baffioni et al. [87].
96. C. Hatzis, "Chronochroma", *Interface* **8**, No. 2, 73 (1979); "Towards an Endogenous Automated Music", *Interface* **9**, No. 2, 83 (1980).
97. S. Holtzman, "Music as System", *Interface* **7**, No. 4, 173 (1978).
98. S. Holtzman, "A Generative Grammar Definitional Language for Music", *Interface* **9**, No. 1, 1 (1980).
99. K. Jones, "Compositional Applications of Stochastic Processes", *Computer Music Journal* **5**, No. 2, 45 (1981).
100. G. Kendall, "Composing from a Geometric Model: *Five Leaf Rose*", *Computer Music Journal* **5**, No. 4, 66 (1981).
101. Baffioni et al. [87].
102. See Christopher Yavelow, "MIDI and the Apple Macintosh", *Computer Music Journal* **10**, No. 3, 11 (1986), which includes a brief description of Levitt's KIT.
103. D. Lorrain, "A Panoply of Stochastic 'Cannons'", *Computer Music Journal* **4**, No. 1, 53 (1980).
104. Hiller and Ames [2].
105. C. Roads, *prototype* (University of California at San Diego, 1975).
106. C. Roads, *A Systems Approach to Composition* (Honors thesis, University of California at San Diego, 1976).
107. C. Roads, *Composing Grammars*, 2nd Ed. (Computer Music Association, 1978).
108. C. Roads, "Interactive Orchestration Based on Score Analysis", *Proceedings of the 1982 International Computer Music Conference* (Computer Music Association, 1982).
109. D. Rothenberg, "A Nonprocedural Language for Musical Composition", *Proceedings of the Second Annual Music Computation Conference* (1975).
110. W. Schottstaedt, "PLA: A Composer's Idea of a Language", *Computer Music Journal* **7**, No. 1, 11 (1983).
111. S. Tipei, "MP1—A Computer Program for Music Composition", *Proceedings of the Second Music Computation Conference* (1975); "Solving Specific Compositional Problems with MP1", *Proceedings of the 1981 International Computer Music Conference* (Computer Music Association, 1981); "*Maiden Voyages*—A Score Produced with MP1", *Proceedings of the 1985 International Computer Music Conference* (Computer Music Association, 1985).
112. C. Ames, "*Crystals*: Recursive Structures in Automated Composition", *Computer Music Journal* **6**, No. 3, 46 (1982).
113. C. Ames, "*Protocol*: Motivation, Design, and Production of a Composition for Solo Piano", *Interface* **11**, No. 4, 213 (1982).
114. C. Ames, "Stylistic Automata in *Gradient*", *Computer Music Journal* **7**, No. 4, 45 (1983).
115. C. Ames, "Notes on *Undulant*", *Interface* **12**, No. 3, 505 (1983).
116. C. Ames, "Applications of Linked Data Structures to Automated Composition", *Proceedings of the 1985 International Computer Music Conference* (Computer Music Association, 1985).
117. C. Ames, "Two Pieces for Amplified Guitar", *Interface* **15**, No. 1, 35 (1986).
118. C. Ames, "A.I. in Music", *Encyclopedia of Artificial Intelligence*, Stuart Shapiro, ed. (New York: John Wiley and Sons, in press).
119. Hiller and Ames [2].
120. C. Ames, "*Concurrence*", *Interface* (in press).
121. Roads [105].
122. Roads [106].
123. Roads [107].
124. Holtzman [98].
125. Jones [99].
126. Ames [112].
127. Dodge [92].
128. Ames [114].
129. S. Gill, "A Technique for the Composition of Music in a Computer", *The Computer Journal* **6**, No. 2, 129 (1963).
130. K. Ebcioğlu, "Computer Counterpoint", *Proceedings of the 1980 International Computer Music Conference* (Computer Music Association, 1980).
131. K. Ebcioğlu, "An Expert System for Schenkerian Synthesis of Chorales in the Style of J.S. Bach", *Proceedings of the 1984 International Computer Music Conference* (Computer Music Association, 1984).
132. M.T. Thomas, "VIVACE: A Rule-Based A.I. System for Composition", *Proceedings of the 1985 International Computer Music Conference* (Computer Music Association, 1985).
133. Ames [113].
134. Ames [114-118, 120]; Hiller and Ames [2].
135. Ames [118, 120]; Hiller and Ames [2].
136. Ames [117]; Hiller and Ames [2].

137. Ames [120].
138. Roads [108].
139. Yavelow [102].
140. Ebcioğlu [131].
141. Barlow [89].
142. Ames [113–117, 120].
143. Koenig [40].

## GLOSSARY

**algorithm**—a computational procedure which, after a finite (but possibly very large) number of steps, *always* returns either a correct solution or the answer that such a solution does not exist.

**Artificial Intelligence**—emulation by computer of human problem-solving activities. An outgrowth of automatic chess-playing procedures proposed by Shannon in 1950 and of theorem-proving programs developed shortly thereafter by Simon, Newell and Shaw in response to Shannon's proposals.

**automated composition**—any use of computer programs that undertake decisions affecting the content of a musical composition.

**Cantor Set**—invented in the course of Cantor's mathematical investigations of infinity, a Cantor Set may be derived from an unbroken segment of the real-number line as follows: divide the segment into three equal portions, exclude the middle portion, and apply the same procedure of division and exclusion recursively to each remaining portion. One might surmise intuitively that such a set would dwindle away to nothing; however, Cantor proved intuition wrong: his set in fact contains a unique counterpart for every number on the infinite real-number line. Cf. R. Courant and H. Robbins, *What Is Mathematics?* 4th Ed. (New York: Oxford University Press, 1947) pp. 248–249.

**formal grammar**—refers to the theory of "linguistic competence" developed during the 1950s by Chomsky. Chomsky divides grammars into two main classes: "phrase-structure grammars", which are sets of top-down rules determining how the whole of a statement relates to its parts (its phrases, clauses, etc.), and "transformational grammars", which are sets of procedures for modifying phrase structures in order to produce well-defined changes in 'meaning' (e.g. mild transformations from active voice to passive, from first person to third, from present tense to past, or more radical transformations such as negation, logical inference, and so on). Because such rules and procedures are typically independent of the specific content of a statement, Chomsky was able to represent them as algebraic manipulations—hence the qualification, "formal".

**frame**—a 'frozen' moment of time; the analogy is drawn from cinematography. An important instance within the context of this article is the *statistical frame*, which denotes a segment of music over which a 'momentary' statistical distribution is realized by a composing program.

**Gestalt psychology**—the study of the perceptual factors affecting how elements of the environment are combined mentally into larger forms ('gestalts'). An outgrowth of commonsense observations made in Wertheimer's 1921 article "Laws of Organization in Perceptual Forms", and a precursor to the modern discipline of cognitive psychology.

**heurism** (or **heuristic**)—a computational procedure which "proceeds along empirical lines, using rules of thumb, to find solutions or answers" (from Webster's Dictionary). Heurisms are less precise than *algorithms*, but they sometimes have the advantage of being usable under less-than-optimal conditions: where an algorithm must always either return a correct solution or fail, a heurism can return a solution which is mostly correct, but which might deviate occasionally from a few less-than-critical specifications.

**hierarchy**—a structure in which the inter-relationships resemble the chain of authority among priests in an organized religion (*hieros* means 'sacred'); each entity in a hierarchy may have any number of 'inferiors' but only one direct 'superior'. Hierarchies are often referred to by computer scientists as *tree structures*.

**left-to-right**—refers to a strategy of handling tasks as they arise, as contrasted with top-down and other strategies that might defer less urgent tasks.

**list**—a structure in which each element has at most one 'predecessor' and one 'successor'.

**Markov chain**—a 'left-to-right' chain of events in which the outcome of each individual event is conditioned by the outcome of its immediate predecessor.

**network**—a structure in which each element can have any number of direct relationships to any number of other elements. Networks are also known among computer scientists as *directed graphs*. A good example of a network is the structure developed by Roget for his 1852 *Thesaurus*.

**paradigm**—a pattern, example or model, typically of a 'classic', 'shining' or particularly insightful character. For instance, a paradigm in a discussion involving several basic premises would be an example that reveals the interplay between most or all of these premises.

**Peano Curve**—derived by infinite recursion, the Peano Curve is a geometric abstraction which defies many of our most basic intuitions concerning how a 'curve' should behave. It has the following properties: every point in the curve resides at the corner of a right angle, and the 'loops' of the curve are infinitely dense in any region (no matter how small) where the curve is defined. Yet the Peano Curve never crosses itself! Cf. E. Kramer, *The Nature and Growth of Modern Mathematics* (Princeton, NJ: Princeton University Press, 1982) pp. 528–529.

**prioritization**—a procedure that ranks a set of objects or tasks in order of preference or urgency.

**random**—refers to a process the outcomes of which are unpredictable or patternless over the short term.

**recursion**—the ability of a process to act upon its own results. Recursive techniques make it possible to implement computational processes that are able, in effect, to propagate 'copies' of themselves. Such processes can spawn multiple sub-processes; these in turn can spawn sub-sub-processes, and so on to arbitrary levels of complexity.

**statistical distribution**—the relative amounts by which differing types occur within a population of elements.

**stochastic**—literally synonymous with 'random'; however, discussions of 'stochastic' processes (as in Xenakis's 1971 book [7]) typically reflect a concern for the large-scale distributions of outcomes.

**top-down**—refers to a process that deals first with generalities and which utilizes the 'insights' gained thereby to cope with more specific tasks.

**with replacement/without replacement**—a favorite paradigm among probability theorists is that of blindly drawing colored balls from an urn, where the probabilities are determined by the proportions of balls of particular colors to the total number of balls. Such selection is said to occur *with replacement* if balls are returned directly to the urn after each draw and *without replacement* if newly drawn balls are set aside. Obviously, the distribution of colors is reflected much more accurately over the short term by the latter method.