Information Retrieval and Language Processing
C.A. Montgomery Editor

A Method for Composing Simple Traditional Music by Computer

Gary M. Rader
University of Pennsylvania

A method is described for composing musical rounds by computer. This method uses some music theory plus additional heuristics. Fundamental to the method is a set of productions together with sets of applicability rules and weight rules which operate on the productions deciding when and to what extent they are available for use.

Several rounds generated by the computer implementation of the method are presented. Generally, the resultant music sounds mediocre to the professional although usually pleasing to the layman. It appears that full-blown music theory is not needed for rounds—all the hardware required for structural levels is not necessary for these pieces. The author has tried to address both musicians and computer scientists.

Key Words and Phrases: artificial intelligence, heuristic programming, models of cognitive processes, computer music, computer composition, music theory, formal languages, probabilistic grammars

CR Categories: 3.44, 3.65, 5.23

Introduction

Very little work has been done to date on writing computer programs that "compose" noncontemporary music. An initial attempt was made in 1955 by Hiller and Isaacson [6] with their "Illiac Suite," a string quartet part of which was based on some elementary rules of music theory. Recently Moorer [8] worked on a method to produce popular melodies. Both of these efforts were based on heuristic processes. There have also been several attempts to generate tonal music by statistical simulation, but this approach does not look promising. This paper presents a method used to mechanically generate traditional, common practice rounds using heuristics modeled on the human composer.

This work was supported by the National Science Foundation under Grant GH-27 and by the US Army Research Office (Durham) under Dept. of the Army Project No. DA-31-124-ARO-D-98.
Author's address: The Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, PA 19104.

Is It Possible?

Almost no one would dispute the statement that it is impossible for a machine to create any aesthetically pleasing piece of art, be it music, painting, or poetry. After all, only a very small number of men in any of these fields have ever been talented enough to be acclaimed great artists. Fundamental to this belief is the fact that art requires both conscious and unconscious evaluations along with an interfacing of the two on the part of both the artist and the perceiver and, as such, is unmechanizable because we cannot even mechanize the conscious part, let alone the whole system.

However, it is possible to talk generally about the meaning of art without going deeply into this complex system of conscious and unconscious evaluations. The cultural anthropologist Gregory Bateson [1] points out that meaning in the context of art can be thought of as being roughly synonymous with pattern or redundancy. For instance, a song can be said to have meaning if it can be separated into two connected parts so that an observer, upon hearing the first, has a better than random chance of guessing the continuation correctly. In music, meaning usually exists on several different levels. That is, most music contains different levels of patterns (i.e. patterns of patterns, patterns of patterns of patterns, and so forth). Thus, if we can somehow formally define several levels of culturally relevant patterns, it is not entirely unreasonable to expect that a computer might be able to compose fairly pleasant music.

Long Range Goal

The problem is not whether computers can compose music, but how far can we go in formalizing human symbolic systems—in this case music. The goal here is not to make aesthetically perfect music, but to make it indistinguishable to the human ear from human-produced music.

With simple forms of music such as rounds, much of the multileveled structure of more complex forms of music need not be dealt with. The object of the present work is to understand better some of the simpler aspects of music and how they mutually relate. By examining the parts and their interrelationships, we gain greater explanatory power of the whole. Once we are largely

able to formalize the musical parameters, melody, harmony, and rhythm, at the microlevel and are able to interrelate them in a common system, we will be able to begin worrying about higher level structural considerations.

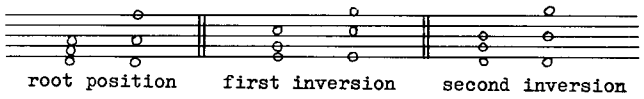
Music Fundamentals

A sequence of single notes is called a line (or part). Several lines may be played at the same time. One line, usually the highest, is called the melody. Two notes which are played at a given point in time constitute a harmony. A melodic (or horizontal) interval is the interval between two successive notes in a line. A harmonic (or vertical) interval is the interval between two notes which are played at the same time. The name of the interval between two notes is found by counting the number of lines and spaces included by the two notes. (The interval between two notes which are one step apart is a second. A third is an interval of two steps, and so on. A unison is an interval of zero steps.)

Harmonies (harmonic intervals) are either consonant or dissonant. The consonant intervals are: unison, third, fourth, fifth, sixth, and octave. Dissonant intervals: second, seventh. Two lines are said to harmonize when all the harmonic intervals are consonant.

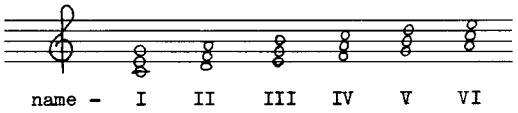
The combination of two or more harmonic (vertical) intervals makes a chord. The simplest type of chord is the triad, composed of two superimposed thirds (or their octaves). Its members are called the root, third, and fifth from bottom to top, respectively. A triad with its root as the lowest note is said to be in root position. A triad with its third as the lowest note is said to be in first inversion, and a triad with its fifth as the lowest note is said to be in second inversion (Figure 1).

Fig. 1.



For our work we will be mainly concerned with the following chords and their inversions (Figure 2). Chords **I, II, III, IV, V,** and **VI** are called the tonic, supertonic, mediant, subdominant, dominant, and submediant,

Fig. 2.

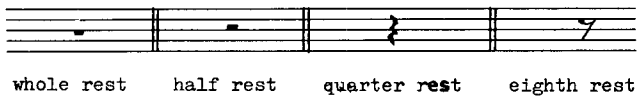


respectively. In practice only certain chord progressions are ever used by composers. Those mainly used are given later in the finite state graph of Figure 5.

The term conjunct (or stepwise) motion refers to a melodic interval of a second (one step); disjunct motion refers to a melodic interval larger than a step (a leap or a jump). In general, a good melodic line contains mostly conjunct motion, with disjunct motion used cautiously for variety.

Points in a line where no tone is sounding are called rests. Rests occur in different durations just as notes do (Figure 3).

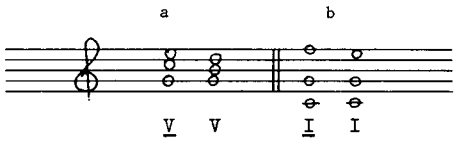
Fig. 3.



One or more notes in a chord may be dissonant (forming a dissonant harmony with some note in the chord, usually the lowest note). The dissonant note in a chord may move to a consonant note while the other notes remain the same; this motion is called resolution and is the source of all harmonic structure. An appoggiatura is a dissonant note which resolves by movement of a step while the nondissonant notes remain stationary. It occurs on a strong beat and its resolution on a weak beat. (In 4/4 meter beats 1 and 3 of each measure are strong while beats 2 and 4 are weak; 1 is stronger than 3 and 2 is stronger than 4.)

The root and third of any second inversion triad are appoggiaturas which generally resolve by moving a step downwards. We shall use the term "V appoggiatura" (and write $\underline{V}$) to describe cases such as the first below where the appoggiatura resolves in a **V** chord. Similarly, we shall use the term "I appoggiatura" ($\underline{I}$) where the resolution is a **I** chord as in the second case below. Thus the harmonic progression in Figure 4 (a) would be **V** appoggiatura to **V** rather than **I** to **V**.

Fig. 4.



Rounds

A round is a circle canon such as Frere Jacques involving the sequential entrance of a number of parts with each strictly imitating the first only later in time. The amount of time between each successive entrance equals the length of the round (number of measures) divided by the number of parts. When a part finishes, it may begin again. To end, we assume that all parts stop simultaneously just prior to an entrance point.

The reason we shall work explicitly only with rounds is this: owing to their relative shortness compared to

other forms and to their built-in melodic, rhythmic, and harmonic imitation on account of repetition, rounds need not have a highly structured melody line or complex harmonic and rhythmic textures. Yet they do require an intelligent handling of these factors. However, they also require the handling of a feature not usually found in other forms. This is the obvious requirement that the melody harmonize with itself (that is, with temporal displacements of itself).

Suppose a round has n parts, where n is some positive integer. We say that this n-part round is in normal form (standard form) if it is written out in n connected staves where each succeeding staff corresponds to the entrance of the next part.

The Basic Method, Informally

The generation process is divided into two parts. Roughly speaking, starting with an empty set of staves in normal form, we first generate a harmonic framework over these staves. Then the melody is generated within this harmonic context.

The methods for generating the harmonic outline and the melody are similar. Both are based on sets of rules stating how chords or notes may be put together. The basic method is iterative and contains a set of "productions" each of which specifies a single choice for the next chord or note. It also contains a set of "applicability rules" which determine when productions may or may not be used. If at least one applicability rule specifies that a production is presently ineligible, it may not be used. The method contains a third set of "weight rules" which indicate the likelihood that an "applicable" production is actually applied by associating a weight with each rule. The weights are variable and may be reset prior to the initiation of any generation. These latter two sets may be viewed as sets of metarules operating on the set of productions. In general, their effects will change after each application of a production.

An example of an applicability rule would be a parallel fifth rule which would not allow the application of any productions yielding parallel fifths (two adjacent occurrences of the interval of a fifth between the same two parts). A weight rule might assign higher weights to rules resulting in stepwise rather than disjunct melodic motion. The effects of both sets of rules are dependent on what has occurred previously during a generation. If an applicability rule determines that a production may not be used, the weight of the production is multiplied by 0, otherwise by 1. A random number generator determines the sequence of applications of productions based on the weightings. A generation ends when all weights are simultaneously zero.

Harmony Generation

By harmonic framework we mean the sequence of chords that will occur on each beat of the round when written in normal form. This harmony does not include an indication of the actual positions of the chords (e.g. root, first inversion, or second inversion for triads). This will be determined later by the melody generator. This harmonic framework will be called a chord pattern.

A chord pattern shall usually consist of those triads closely related to the key of C major. The number of chords in a chord pattern equals the number of beats that any staff of a round in normal form contains.

The harmony generator consists of the following:

Productions:

H1 **I** may follow any sequence of chords (even the null sequence).

H2–H5 A chord may be followed by the chord a third below, a second above, a fourth above or by itself, respectively.

H6 **I** and **IV** may be followed by $\underline{\mathbf{V}}$ and $\underline{\mathbf{I}}$, respectively.

H7 $\underline{\mathbf{V}}$ and $\underline{\mathbf{I}}$ may be followed by **V** and **I**, respectively.

H8 **I** may precede any chord except $\underline{\mathbf{V}}$ and $\underline{\mathbf{I}}$.

Applicability rules:

H9 No rule is applicable after the chord pattern has reached an initially specified length.

H10 The first chord must be **I**.

H11 The last two chords must be **I**.

H12 $\underline{\mathbf{I}}$ and $\underline{\mathbf{V}}$ cannot occur on the second and fourth beats.

H13 **IV, I,** and **IV** may not follow **VI, III,** and **III,** respectively.

Weight rules:

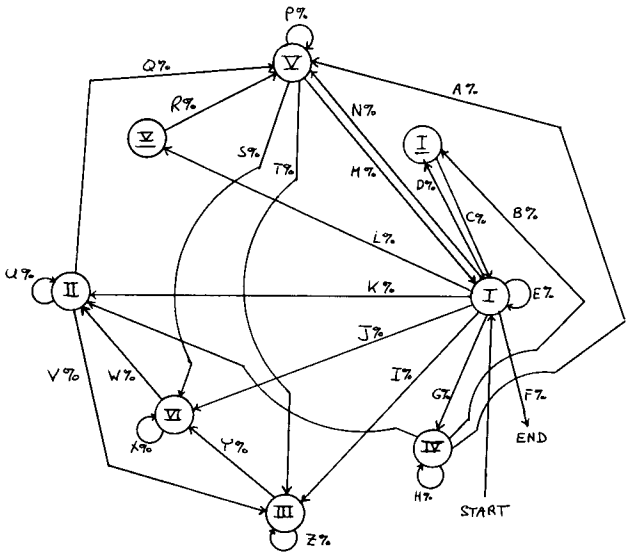
There is one weight rule for each production, which assigns a fixed weight to that rule.

We assume a meter of 4/4. H1 is used only to obtain the first chord in a chord pattern. H7 is used whenever possible (insuring resolution). The weights are usually set so that H4 or H5 is usually applied, H6 or H8 is sometimes applied, and H2 or H3 is occasionally applied. The finite state graph below describes all possible chord patterns the harmony generator can produce when the arrows are weighted appropriately.

H1–H8 derive in part from Piston [9] and Vauclain [14] and in part from my own musical training, H9–H11 are my own inventions, and H12–H13 are extracted from Vauclain. Other sets of rules might do just as well or even better. The relationships among the weights in the weight rules are fairly crucial.

To read Figure 5 we must follow the arrows beginning at START. Each time we come to a circle (a state), we write down the chord contained in it. If more than one arrow leads from a circle, one must be chosen and followed; the choice depends on the corresponding percentage. The chord pattern is completed when we come to END. The percentages attached to the arrows emanating from each state generally change while traversing the graph but always total 100 percent. For example,

Fig. 5.



A + B + H + O = 100 percent always but B usually alternates between zero and a nonzero percentage due to Rule H12. More precisely, the transition function between states is a random function with a distribution so chosen that the probability of obtaining nontraditional harmonic progressions is zero.

However, as it now stands, the harmony generator will not necessarily give a chord pattern of an initially specified length while observing the applicability rules without some sort of look ahead. To remedy this, we reverse the order of generation so that a chord pattern is produced from right-to-left beginning with the last chord. Then, since I can precede any chord except V, and since V cannot occur on the second beat of a measure, we can always end by choosing I to begin the appropriate measure.

Melody Generation

The melody generator utilizes the same basic method in producing notes for a round. It will generate these notes within the constraints of a chord pattern and according to practiced rules of music and a collection of heuristic rules of melody.

Allowable notes will consist of all notes from the g below middle c through the g two octaves above it plus the rest, which may be considered as a null note. These notes will have a fixed duration of an eighth note. To obtain notes with longer durations, we add a new character whose meaning will be to lengthen the duration of any note it follows by an eighth note. In this way notes with any duration which is an integral multiple of eighth notes can be generated.

To generate a round, the melody generator must be initially supplied with a chord pattern, the number of parts, the maximum allowable number of octave jumps, and the maximum allowable spread between the highest and lowest notes in the completed round. The resultant round will have a length (ln) equal to the length of the chord pattern times the number of parts. The round is generated in normal form, top to bottom, left to right.

Productions:

M1–M2 An eighth note on middle c or an octave above may be added to any sequence of notes (even the null sequence).

M3 The last note may be increased in duration by a quarter note.

M4–M13 The last note may be increased in duration by an eighth note and followed by an eighth note which is 1, 2, 3, 4, or 5 steps above or below the last "stepwise note" (see M37).

M14–M19 The last note may be increased in duration by an eighth note and followed by an eighth stepwise note 1, 2, or 7 steps above or below the last stepwise note, respectively.

M20–M21 The last note may be followed by two eighth stepwise notes moving stepwise up or down from the last stepwise note.

M22–M23 The last note may be followed by two eighth notes moving up (down) one step and then back down (up) one step from the last stepwise note, respectively.

M24–M25 The last note may be increased in duration by an eighth note and followed by an eighth rest or an eighth note in the same position, respectively.

M26 The last note may be increased in duration by an eighth note.

Applicability rules:

M27 No rule is applicable after the length of the round equals ln.

M28 M1 or M2 must be applied initially and never again.

M29 M26 must be the last production applied and never before.

M30 A production resulting in parallel unisons, fifths, octaves, or twelfths from the first half of a beat to the second half of the same beat or from the second half of a beat to the first half of the next beat may not be applied.

M31 A production yielding a note which is not a member of the chord corresponding to the present beat may not be applied unless it is a rest.

M32 A production yielding the harmonic interval of a fourth may not be applied unless it resolves or occurs in a first inversion triad or off the beat.

M33 If the previous chord was an appoggiatura, the present note must be a rest, the same as the last note, or move 1 step down.

M34 The maximum spread between the highest and lowest notes of the round may not exceed an initially specified amount.

M35 Half and dotted quarter notes may only occur on the first and third beats.

M36 The number of octave jumps may not exceed an initially specified number.

M37 (Melodic Rule) Beginning with the first note, the melody must progress stepwise or, if it jumps, it must either return and continue the stepwise movement from the point at which it jumped or move back toward this point until it does continue the stepwise motion. In the case of appoggiaturas, the melody may first resolve before satisfying the above. An octave jump displaces the stepwise motion by a similar amount. An octave jump may occur only when the last note was a stepwise note (part of the stepwise motion). If continuing by two stepwise eighth notes (M20 or M21) would result in unallowed parallel motion, the melody may progress by the interval of a stepwise third (M16 or M17 respectively).

M38 An octave jump must be followed by movement in the opposite direction.

M39 Only the root of an appoggiatura chord may be doubled.

M40 M24 may be applied only if no other rule is applicable.

M41 The root of an appoggiatura chord and its resolution must be either a half note or two quarter notes.

M42 No note outside of the range from the g below middle c through the g two octaves above may be used.

Weight rules:

There are two weight rules for each production. One is used if applying the production will double a note (up to an octave); the other is used if no doubling will occur. The weights are normally set so that doubling is fairly unlikely and so that stepwise motion is much more likely than disjunct motion. We have the following additional rules.

M43 If the last note was a leading tone (b in the key of C), the likelihood that the next note is the root one step up is increased by an initially specified amount.

M44 The likelihood of a note being doubled is increased by an initially specified amount if that note is a c, f, or g.

M45 The likelihood that the last note is c is increased by an initially specified amount.

M1–M26 are based on a combination of Vauclain [14], my own musical training, and a study I undertook of rounds and canons. M27–M43 and M45 derive from a combination of my own musical training, my imagination, and what appeared to be necessary to avoid the gross musical "errors" of earlier versions of this set. M44 comes from Vauclain. Again, there are indubitably other formulations of these sets of rules that would do just as well or better than the present ones. And again, the relationships among the weight settings of the weight rules are fairly crucial.

Formal Description

We now give a formal description of the basic method used in the composing process under the guise of a stochastic grammar.

A stochastic grammar is similar in its basic structure to a Chomsky grammar. It contains a terminal and a nonterminal alphabet, a set of productions, and a starting symbol. The difference lies in the fact that it also contains a set of probability functions and a set of input information. The probability functions are used to compute how probable the application of each production is at every stage in the generation process. One function is associated with each production. Generally these functions are dependent on the input information and on the past sequence of applications of productions in the current generation. (From this fact comes the name "stochastic.") The input information is a set of data supplied to the grammar before the start of a generation.

For convenience we include two more sets, a generic alphabet and a set of alphabet functions. They are not necessary but are used in production schemata to make the grammar's structure more compact and transparent. (They are also used in the computer programs to make them more efficient.) Each character in the generic alphabet represents all the members of a specified subset of the complete alphabet. When one appears in a production, it will stand as an abbreviation for the entire class of productions obtained by replacing the generic character by each member of the subset. However, if the same generic character appears more than once in a production, all occurrences must be replaced by the same member. Alphabet functions are functions from the complete alphabet into itself. As such, they may include suitable generic characters in their domains. Finally, the grammar contains a bijection which associates a probability function with each production.

A schotastic grammar $\mathcal{G}$ is a system of nine types of elements

$\mathcal{G} = (\Sigma, G, V, R, S, I, F, P, M)$ where:

Σ is the terminal alphabet.

G is the generic alphabet.

V is the complete alphabet.

R is the set of productions.

S is the starting symbol.

I is the set of input information.

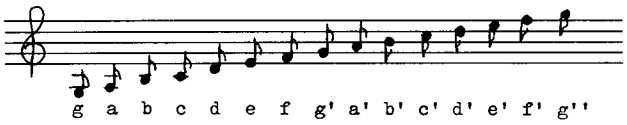
F is the set of alphabet functions.

P is the set of probability functions.

M is a bijection $M : R \rightarrow P$.

Each $p \in P$ is assumed to yield only one value at a given moment. This value is initially computed before any productions are applied and then recomputed after each application of a production. The set I usually includes a probability constant for each $p \in P$. Unless otherwise stated, each generic character $g \in G$ will be considered generic over the alphabet $V - G$. We

Fig. 6.



do not restrict the usage of terminal or generic characters on the left hand side of productions. (For example, we allow $x \rightarrow aa$ and $a \rightarrow aa$ for $x \in G$, $a \in \Sigma$.) A derivation is complete when all probability functions are zero at the same instant.

The stochastic grammar is the same as the general method explained previously. The sets of applicability and weight rules are now subsumed by P.

Formal Specification of the Harmony Generator

The following stochastic grammar $\mathcal{G}_H$ will determine the harmony or basic vertical structure of the rounds by outputting chord patterns. Let

$\mathcal{G}_H = (\Sigma_H, G, V, R, S, I_H, F, P, M)$ where:

Σ_H = **(I, II, III, IV, V, VI, I, V)**.

$G = \{x\}$.

$V = \Sigma_H \cup \{S\}$.

$R = \{(1)S \rightarrow \mathbf{I\ I},\ (i)x\ \alpha \rightarrow f_i(x)x\ \alpha$ for $2 \leq i \leq 8$, $\alpha \in V^+\}$.

S is the starting symbol.

$I_H = \{\langle\text{length}\rangle, (p_1, p_2, \ldots, p_8)\}$.

$F = \{f_i : \Sigma \rightarrow \Sigma$ for $2 \leq i \leq 8$ where $f_2 : \sigma \rightarrow D^{-1}$ (max $\{1$, (max $\{D(\sigma)\} - 6)\})$, $f_3 : \sigma \rightarrow D^{-1}$ (max $\{1$, (max $\{D(\sigma)\} - 4)\})$, $f_4 : \sigma \rightarrow D^{-1}$ (min $\{10$, (max $\{D(\sigma)\} + 2)\})$, $f_5 : \sigma \rightarrow \sigma$, $f_6 : \sigma \rightarrow D^{-1}$ (max $\{1$, (max $\{D(\sigma)\} - 3)\})$, $f_7 : \sigma \rightarrow D^{-1}$ (min $\{10$, (max $\{D(\sigma)\} + 1)\})$, $f_8 : \sigma \rightarrow \mathbf{I}$ for $\sigma \in \Sigma$ where D is the relation **(III,** 1), **(VI,** 2), **(VI,** 3), **(II,** 4), **(II,** 5), **(V,** 6), **(V,** 7), **(I,** 8), **(I,** 9), **(IV,** 10)$\}$.

$P = \{(1)\ p_1 \times (\text{ln} = 0)$, $(2)\ p_2 \times g \times n \times (0 < \max\{D(h)\} - 6 < 11)$, $(3)\ p_3 \times g \times n \times (0 < \max\{D(h)\} - 4 < 11)$, $(4)\ p_4 \times g \times n \times (0 < \max\{D(h)\} + 2 < 11)$, $(5)\ p_5 \times g \times n$, $(6)\ p_6 \times g \times n \times (0 < \max\{D(h)\} - 3 < 11)$, $(7)\ p_7 \times g \times (\neg n) \times (\text{ln} \equiv 1 \pmod 2) \times (0 < \max\{D(h)\} + 1 < 11)$, $(8)\ p_8 \times (0 \neq (g \times a + (\text{ln} = \langle\text{length}\rangle - 1)))$ where ln is defined to be the length of the partial derivation (the amount of derivation thus far completed), g is defined to be 1 if $0 < \text{ln} < \langle\text{length}\rangle - 1$; 0 otherwise, h is defined to be the leftmost element of the partial derivation with **I** catenated on the right, n is defined to be 1 if $h \notin \{\mathbf{I}, \mathbf{V}\}$; 0 otherwise, and a boolean value of true is defined to be 1, of false, 0$\}$.

After each calculation of the values of the above set, we must divide each by their sum (if not zero) so that they total 1.

M is defined by M : Production $(i) \rightarrow$ Probability function (i) for $1 \leq i \leq 8$.

Formal Specification of the Melody Generator

The stochastic grammar $\mathcal{G}_M$ where output is the actual sequence of notes encoded is a little more complex than $\mathcal{G}_H$. The notes are labeled g, a, ..., f, g', a', ..., f', g'' beginning with the g below middle c up through the g two octaves above (see Figure 6).

Each note will have the duration of an eighth note. In addition, an eighth note rest will be labeled r, and the letter n will stand for an extension of one eighth note to the duration of the previous note or rest. For example, c n n n r n represents a half note middle c followed by a quarter note rest.

For simplicity, we assume the notes (g through g'') are ordered as shown in Figure 6 from left to right. The functions

R_j and $L_j : \Sigma_M - \{\text{r, n}\} \rightarrow \mathbf{V}$ are defined by

$R_j : \sigma \rightarrow$ to the jth note to the right of σ if it exists, otherwise to g

$L_j : \sigma \rightarrow$ to the jth note to the left of σ if it exists, otherwise to g.

These functions facilitate the defining of the alphabet functions for $\mathcal{G}_M$. In the case where the jth note to the right or left, respectively, does not exist, we do not care about the actual assignment as the probability function attached to this function will have a value of zero.

$\mathcal{G}_M$ is defined to be

$\mathcal{G}_M = (\Sigma_M, G, V, R, S, I_M, F, P, M)$ where:

$\Sigma_M = \{\text{g, a, b, ..., f, g', a', ..., f', g'', r, n}\}$.

$G = \{x\}$ where x is generic over $V - (\Sigma_M \cup \{S\})$.

$V = \Sigma_M \cup \{S$, [g], [a], [b], ..., [f], [g'], [a'], ..., [f'], [g'']$\}$.

$R = \{(1)\ S \rightarrow$ [c], $(2)\ S \rightarrow$ [c'], $(3)\ \alpha x \rightarrow \alpha x$nn, (i) $\alpha x\beta \rightarrow \alpha x\beta f_i(\text{U}(x))$, $(26)\ \alpha \rightarrow \alpha$n $(4 \leq i \leq 25$; $\alpha \in V^*, \beta \in \Sigma^*_M)\}$ where $U : V - (\Sigma_M \cup \{S\}) \rightarrow V$ by $\text{U} : [v] \rightarrow v$ (see F).

Productions 1–26 here correspond to M1–M26 given previously.

S is the starting symbol.

$I_M = \{\langle\text{length}\rangle, \langle\text{number of parts}\rangle, \langle\text{chord pattern}\rangle, \langle\text{spread}\rangle, (p_1, p_2, \ldots, p_{26}), \langle\text{extra probabilities for weight rules}\rangle\}$.

$F = \{\text{f}_i : \Sigma_M - \{\text{n, r}\} \rightarrow \mathbf{V}$ for $4 \leq \text{i} \leq 25$ by $\text{f}_4 : \sigma \rightarrow \text{n} \frown R_1(\sigma)$, $\text{f}_5 : \sigma \rightarrow \text{n} \frown L_1(\sigma)$, $\text{f}_6 : \sigma \rightarrow \text{n} \frown R_2(\sigma)$, $\text{f}_7 : \sigma \rightarrow \text{n} \frown L_2(\sigma)$, $\text{f}_8 : \sigma \rightarrow \text{n} \frown R_3(\sigma)$, $\text{f}_9 : \sigma \rightarrow \text{n} \frown L_3(\sigma)$, $\text{f}_{10} : \sigma \rightarrow \text{n} \frown R_4(\sigma)$, $\text{f}_{11} : \sigma \rightarrow \text{n} \frown L_4(\sigma)$, $\text{f}_{12} : \sigma \rightarrow \text{n} \frown R_5(\sigma)$, $\text{f}_{13} : \sigma \rightarrow \text{n} \frown L_5(\sigma)$, $\text{f}_{14} : \sigma \rightarrow \text{n} \frown [R_1(\sigma)]$, $\text{f}_{15} : \sigma \rightarrow \text{n} \frown [L_1(\sigma)]$, $\text{f}_{16} : \sigma \rightarrow \text{n} \frown [R_2(\sigma)]$, $\text{f}_{17} : \sigma \rightarrow \text{n} \frown L_2(\sigma)]$, $\text{f}_{18} : \sigma \rightarrow \text{n} \frown [R_7(\sigma)]$, $\text{f}_{19} : \sigma \rightarrow \text{n} \frown [L_7(\sigma)]$, $\text{f}_{20} : \sigma \rightarrow [R_1(\sigma)] \frown [R_2(\sigma)]$, $\text{f}_{21} : \sigma \rightarrow [L_1(\sigma)] \frown [L_2(\sigma)]$, $\text{f}_{22} : \sigma \rightarrow R_1(\sigma) \frown \sigma$, $\text{f}_{23} : \sigma \rightarrow L_1(\sigma) \frown \sigma$, $\text{f}_{24} : \sigma \rightarrow \text{n} \frown \text{r}$, $\text{f}_{25} : \sigma \rightarrow \text{n} \frown \sigma\}$,

where "$\frown$" represents catenation. The square-bracketed characters indicate stepwise melodic progression.

Since P is rather extensive, it is not given explicitly here. In effect, P is just a mathematization of rules M27–M45 given earlier.

M is defined by M : Production $(i) \rightarrow$ Probability function (i) $(1 \leq i \leq 26)$.

To Obtain a Round

Now to obtain a round, we supply $\mathcal{G}_H$ with the desired length divided by the number of parts, reset the initial weights if we wish, and initiate $\mathcal{G}_H$. This gives us a chord pattern. We then supply $\mathcal{G}_M$ with this chord pattern and the number of parts after resetting any weights that we want. $\mathcal{G}_M$ will generate a sequence of encoded notes, to which we apply U ($U(m_1, \ldots, m_p) \equiv U(m_1) \cdots U(m_p)$) and then rectangularize. (U was defined in R of $\mathcal{G}_M$. By rectangularize, we mean reshape the vector of notes into its normal form. For a 3-part, 12 measure round, rectangularization would map measures 1–4 to the first staff, measures 5–8 to the second staff, and measures 9–12 to the third.)

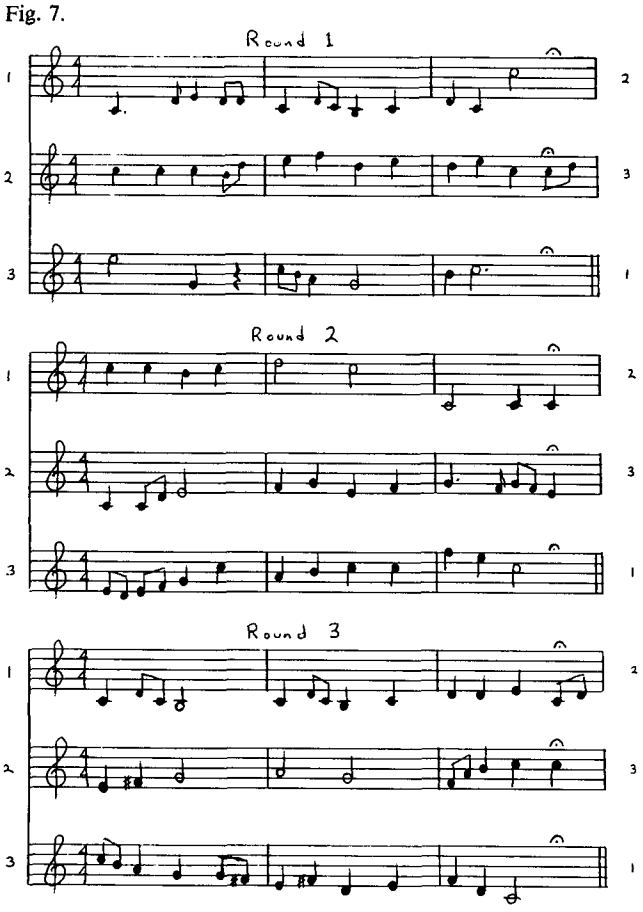
Some Examples

The APL embodiment of the above process (running on an IBM 370/165) has so far generated over 50 short 2- to 5-part rounds. The concentration has been on 3-part rounds as they are the most difficult for humans to compose. (However, 4/4 is the easiest meter to compose in.) In Figure 7 appear three of the more interesting of these. The pause (𝄐) designates the note to be held when ending. For all, the maximum allowed spread was 11, the maximum number of octave jumps was 1, the cfg doubling weight (see M44) was 10, and the last-note-equals-c weight was 1000. The first round has weights of 1, 1, 1, 12, 4, 6, 8, and 6 for productions H1–H8, respectively. Its chord pattern is **I I V V I II V I V I I I.** The nondoubling weights for M1–M26 were 1, 1, 20000, 5000, 5000, 80, 80, 40, 40, 40, 40, 10, 10, 64000, 64000, 5000, 5000, 700, 700, 5000, 5000, 0, 0, 1, 5000, and 1, while the doubling weights were 1, 1, 200, 0, 0, 10, 10, 10, 10, 10, 10, 10, 10, 320, 320, 500, 500, 10, 10, 500, 500, 0, 0, 1, 100, and 1, respectively.

For the second round, the weight for H5 was changed to 3, the nondoubled weights for M4, M5, M18, M19, M22, and M23 became 20000, 20000, 400, 400, 2500, and 2500, respectively, and the doubled weights for M6–M13, M16, M17, M20, and M21 became 40, 40, 20, 20, 3, 3, 1, 1, 50, 50, 50, and 50, respectively. Its chord pattern is **I I III VI II V I IV I I I I.**

The third round is more experimental, utilizing a "key pattern" which is similar to a chord pattern but controls key modulations instead of harmony by defining a key for each vertical beat. The key pattern is CGGGGGGGCCCC and the chord pattern **I II V V VI II V I II V I I.** Because the interfacing between the generators of these patterns has not been programmed, both patterns were human generated to insure some correspondence between them. All other relevant weights remained unchanged except that M22 and M23 were eliminated. Each round required something under 6 seconds of cpu time for generation. Output was pseudo-musical score to facilitate evaluation.

Fig. 7.



Evaluation

The best way to evaluate this music is to listen to it. Performances should be by instruments or voices of different tone qualities to allow a following of the melody line but all in the same range to prevent unwritten harmonic inversions.

Under the present rules, minor deviations from accepted harmonic practice (such as simultaneous leaps in similar motion to the same note) are still possible, but it appears that these can be eliminated through the addition of appropriate applicability rules. The melody is a much more critical element, being generally musically acceptable but usually not very interesting. Standard melodic features such as arpeggiation and rhythmic and melodic imitation are either impossible or very unlikely under the present rules for generating melodies.

On a higher plane, the use of probabilities in the form of weights to help decide the melodic and harmonic structure might be viewed with alarm. It might be better to use the term "guided probabilities," reflecting the use of applicability rules. Historically, both

Hiller and Moorer used probabilities in their selection procedures. The musicologist Leonard B. Meyer [7, pp. 54–56] states that styles of music are in effect "complex systems of probability relationships" and gives convincing evidence in support of this. He also points out that these systems of probability relationships are explicitly acknowledged at a very general level in the names given to tones in the Western, Chinese, and Indian systems of music. (For example, in Western music the name "leading tone" reflects the fact that it usually leads up to the tonic.)

The tunings (weight settings) used are not necessarily the best. Perhaps better tunings would result in better music. It does appear that certain general styles of music may be obtained through widely different tunings. Also, the set of productions could be expanded.

Conclusion

In conclusion, the program "composes" at a mediocre level though at a generally quite acceptable level for the man on the street. The harmony is often quite good while the melody is usually acceptable but dull. It appears that full-blown music theory is not needed for rounds—all the hardware required for structural levels is not necessary for these pieces.

Acknowledgment. The author wishes to thank Saul Gorn and Constance Vauclain for their continuing advice on the work described here.

Received June 1973; revised May 1974

References

1. Bateson, Gregory. Style, grace, and information in primitive art. In *Steps to an Ecology of Mind.* Ballantine Books, New York, 1972.
2. Brahms, J. *Kanons fur Frauenstimmen.* C.F. Peters, Leipzig, 1891.
3. Gorn, Saul. On the mechanical simulation of habit-forming and learning. *Information and Control 2* (Sept. 1959), 226–259.
4. Gorn, Saul. The computer and information sciences: a new basic discipline. *SIAM Rev. 5* (Apr. 1963), 150–155.
5. Gorn, Saul, The identification of the computer and information sciences: their fundamental semiotic concepts and relationships. *Foundations of Language 4* (Nov. 1968), 339–372.
6. Hiller, L.A., and Isaacson, L.M. *Experimental Music-Composition with an Electronic Computer.* McGraw-Hill, New York, 1959.
7. Meyer, L.B. *Emotion and Meaning in Music.* U. of Chicago Press, Chicago, 1956.
8. Moorer, J.A. Music and computer composition. *Comm. ACM 15,* 2 (Feb. 1972), 104–113.
9. Piston, W. *Harmony.* W. W. Norton, New York, 1941.
10. Rader, G.M. An algorithm for the automatic composition of simple forms of music based on a variation of formal grammars. Moore School Rep. No. 73-09, Philadelphia, 1973.
11. Salomaa, A. Probabilistic and weighted grammars. *Information and Control 15* (Dec. 1969), 529–544.
12. Taylor, M.C. (Ed.). *Rounds and Rounds.* Hargail Music Press, New York, 1959.
13. Taylor, M.C., Windham, M., and Simpson, C. (Eds.). *Catch That Catch Can.* E.C. Schirmer Music Co., Boston.
14. Vauclain, A.C. Music 25. Dept. of Music working paper, U. of Pennsylvania, Philadelphia, Pa., 1971.

Programming Techniques
G. Manacher Editor

Register Allocation Via Usage Counts

R.A. Freiburghouse
Honeywell Information Systems Inc.

This paper introduces the notion of usage counts, shows how usage counts can be developed by algorithms that eliminate redundant computations, and describes how usage counts can provide the basis for register allocation. The paper compares register allocation based on usage counts to other commonly used register allocation techniques, and presents evidence which shows that the usage count technique is significantly better than these other techniques.

Key Words and Phrases: optimization, redundant computations, common subexpressions, register allocation, compilers, programming languages, virtual memory, demand paging

CR Categories: 4.12, 4.2, 4.39

1. Introduction

Algorithms for eliminating redundant computations are well known and widely implemented [1, 4, 5, 8, 11]. Similarly, several techniques for register allocation have appeared in the literature [2, 3, 4, 10, 11, 12]. This paper introduces a very simple mechanism, the usage count, which ties these two subjects together and which provides the basis for an easily implemented technique for register allocation that is remarkably efficient.

Author's address: Honeywell Information Systems Inc., 575 Technology Square, Cambridge, MA 02139.