"Sonic Set Theory" (a.k.a. "Sonic Subsets") was published in the
"Proceedings of the Symposium on Small Computers and the Arts",
IEEE Oct.1982

Sonic Set Theory: A Tonal Music Theory for Computers

Tools for Algorithmic Composition
With an Algorithm for Chord Selection for Dessert
by Laurie Spiegel, New York City, August, 1982

(Published in the "*Proceedings of the Symposium on Small Computers and the Arts*, IEEE October, 1982.)

For millenia, humans have wanted to be able to listen to the music of numbers. Pythagoras, with his "music of the spheres", and Kepler, in his "harmony of the planets" idea are the two intellectual adventurers who have most inspired us to this ideal. With computers, we can at last listen to number as music, create music as number made audible. But despite these long sought and newly arrived pleasures, we also want to use the tools of math and logic to create music meaningful in the same ways that traditional music has been. We wish to do so in order to expand the realm of music in an evolutionary manner, to make the pleasure of it's creation easier and accessible to more people, and to study and better comprehend its effect on us, to heighten our self-awareness and self-understanding.

Algorithmic music

We can separate algorithmic music into four main tendencies. The first of these is the sonic embodiment of extra-musically derived relationships (mathematical, visual, or physical). Kepler's idea, and translations into sound of mathematical series (prime numbers modulo 12, etc.) furnish good examples of this first-mentioned trend.

The second tendency consists of attempts to extend the boundaries of what we know as music, to extend the areas of the imagination and its products which we are able to experience in common. This can be exemplified by Xenakis's stochastic techniques.

The third is the description and composition by rule of structures derived from the study of traditional music, such as Hiller's ILIAC suite. Such work would include simulation programs which might do such things as create melodies against given chords, or conversely, or generate music in a particular style from nothing but rules. One goal here is the development of easier, more economical ways of describing large numbers of musical events.

The fourth trend in algorithmic composition would be the musical analog of artificial intelligence research. Musical "AI" might try to generate all the musical possibilities for a given situation and then "filter" out (by logical constraints) everything that won't "work" musically. Systems of such rules and constraints would evolve, gradually, as embodiments of our self-understanding, and might tie in closely with research in cognition, perception, information theory, communications or game theory. I distinguish these somewhat overlapped third and fourth tendencies chiefly on the basis of their goals, the former predominantly desiring to create repertoire and to understand music as we have known it in the past, and the latter predominantly wishing to add to our self-knowledge and capabilities.

A Computer Music Theory

Music is subtle and mysterious in its ways of working on us, but - up to a point as yet to be approached - it's workings can be comprehended and described. Different periods have had different music theories and conceptual tools, intended to be useful to composers in creating works, and to players and listeners in interpreting them. Species counterpoint, solfeggio, Rameau's theories on harmony and figured bass are among the most significant of such tools for European music. As computers lead us into new realms of creative work in music, providing us with new compositional, elaborative, analytical, and descriptive tools for music, we may be able to benefit from a new model or theory of music which relates better to the concepts we use in working with these new tools than do the music theories of the past.

It is the purpose of this paper to begin to furnish such conceptual tools, to at least scratch the surface of a variant of music theory (tonal to start) which will fit more easily into the conceptual vocabulary of programming than those already in use. This beginning theoretical revision contains little that is new. It consists, instead, of a way of organizing what we already understand so that it will become, hopefully, a bit easier to describe musical relationships in computer terms. The two concepts which it introduces to music are set theory and the idea levels of indirection. (The idea of hierarchy already exists in traditional music theory, particularly in the domain of time.)

Musical Frequency Patterning is Unstepwise

The way that frequency is meaningful in music is not well represented by a liner scheme of organization. You can compose soundpieces within the equal tempered scale by treating it like a uniform, ordered, and unweighted set of integers 1 to 12, but the structures of such works will not fit easily to the way the ear makes sense of music. Music generally moves in units which consist of unequal numbers of scale steps. It moves among recognizable patterns by common tones, common harmonic content of tones, and by intervals which occur within established patterns (such as chords). In addition, some scale members (for example, tonic and dominant) occur with much higher frequency than do others.

In our culture, dominant patterns consisting of unequal intervals include tonal scales, triadic and other chords, and the patterns of movement of the roots of chords (which tend to be in fifths or thirds). An arbitrary mathematically derived number sequence in base 12 may be interesting to the mind, but it will be unlikely to be a highly controllable way to rouse the emotions through the manipulation of expectation in time, or by harmony or melody. Equal-step quantization within the octave is not how our music is set up.

Most music meaningful to us is tonal in its patterning, not chromatic (using all 12 tones within the equal tempered octave). The closest harmonically related chords to C major are not C-sharp or B major, but A minor and G major. Harmonic content (internal frequency ratios), not the proximity of frequencies, is what counts.

The question posed is this: How can we better deal with music's complex non-uniform non-scalar patterning with computer/mathematical logic? By "better" I mean both the abandonment of methods which are ineffectual in a computer language context, and the adoption or optimization of methods and concepts which permit us to make full musical use of the computer's unique capabilities.

Sets and Subsets in Musical Frequency

Tonal pitch collections, such as chords, scales, or modes, can be viewed as templates which can be laid over the equal tempered scale. Such templates as we are about to look at for the tonal/modal collection could be tried (found or invented) for any scale (microtonal, etc.), or any group or cluster of pitches. However, for purposes of simplicity and clarity, we'll restrict the application of this idea in this article to the conventional uses of the conventional equal tempered scale.

A template can also be regarded as a way of referencing a subset, of including and excluding set members by virtue of their relationship with each other rather than on the basis of their specific characteristics. Tonal music can therefore be considered in terms of sets and subsets, maps and submaps, or templates on templates (sequential layers of pitch filters).

On the first level, that of the most inclusive or general set, we posit continuous frequency. Discrete pitch collections can be viewed as subsets of continuous frequency. As we select smaller subsets, we come to the equal tempered scale. The next most meaningful and general subsets include scales and modes (major, harmonic or melodic minor, dorian, phrygian, lydian, mixolydian, aolian, pentatonic, and other more exotic modes). If we're dealing with modal music, we might go to the tetrachord as the next most generally meaningful subset. For tonal music, we go to the triad.

Triads are easy templates which can be dealt with even without a more general tonal template. There are 4 kinds, which are easily stored as arrays of offsets from the tonic of a chord. Once you know the root (or tonic) of the triad, you add these offsets, which are counted in terms of our standard minimal unit, the semitone. A major triad's 2 upper notes are at offsets of 4 and 7 semitones from the tonic, minor at 3 and 7, diminished at 3 and 6, and augmented at 4 and 8. Any of these addends plus a multiple of 12 (to offset the pattern to other octaves) will give you a note that is in the triad.

Here's a simple example of how this idea can be used:

```
REM Arpeggiate chords
REM minor then major on each
REM step of a rising bassline.
REM Works musically by keeping
REM common tones across bass movement.

Dimension Etscale(12 * number of octaves)
REM fill with Equal Tempered scale values
REM for sending to particular hardware

Dimension Major(4), Minor(4), Bass(13)

REM create rising bassline
REM (or you could compose one)
For I = 0 to 12
Bass(I) = I
Next I

Minor(0)=0
Minor(1)=3
Minor(2)=7
Minor(3)=12

Major(0)=0
Major(1)=4
```

```
Major(2)=7
Major(3)=12

REM for each scale degree
For B = 0 to 12

REM arpeggiate minor chord
For I = 0 to 3
Note = Etscale(Bass(B) + Minor(I))
REM play note here
Next I

REM arpeggiate major chord
For I = 0 to 3
Note = Etscale(Bass(B) + Major(I))
REM play note here
Next I

REM move to next bass note
Next B

End
```

Triads aren't sufficient for tonality, though. Chords move by two main methods.

The first type of chordal movement is to another chord whose root or "tonic" is a fifth away from the current one. To create this kind of root movement, we might substitute some kinds of movement through the following array, consisting of a cycle of fifths for our rising baseline in the above example:

```
Dim Cycle5(12)
REM fill it with
REM 0 7 2 9 4 11 6  1  8  3  1 0  5
REM C G D A E B  F# C# G# D# A#  F
```

The second most useful type of chord movement is by common tones, such as to a parallel or relative minor. In both these cases, 2 notes of a triad remain common while the third one changes. C major is "parallel" to C minor, the C and G being common to both while only the E and E-flat differ. C major and A minor are "relative" major and minor chords, the C and E being common to both. Parallel chords are easily interchanged by defining them as dyads a perfect fifth apart, and then algorithmicly filling in the major or minor third of the chord.

Relative major and minor chords can easily be dealt with by using an array which consists of the union of the major and minor triads (A,C,E,G) and simply using a starting index of 1 for major and 0 for minor when reading 3 consecutive offsets from this array. (The array would not have the note names in it, but would contain offsets from the tonic in semitones, which would be usable in any key and for any chord in the key. Use them by adding them to your tonic's offset into the e.t. scale array.) Another way of dealing with relative major and minor chords is by using dyads, sets consisting of 2 notes a third apart, and taking the union of 2 intersecting dyad sets as a triad, but because there are 2 types of "thirds" this is not practical until we have the following tonal set to index into.

Tonality as a Great Simplifier

The above already grows more complex than is desirable. One really wants the simplest, most general, and

most internally consistent scheme of organization possible, so as to think as little as possible about "bookkeeping" and as much as possible about music. Therefore, to most simply do tonal music, we'll start with the tonal pitch collection itself as our basic working set, rather than the 12-tone scale.

The tonal set can be viewed as a subset of, or template for, the 12-tone equal tempered pitch collection. It's internal intervals (offsets between adjacent members of the equal tempered superset) are 2,2,1,2,2,2,(1). The interval of each tonal pitch as an offset from the tonic of the key are 0,2,4,5,7,9,11(,12). These are stored in an array which will be used to index into the 12-tone superset array.

This reduces our basic working set of pitches to 7. (The word "octave" suddenly has meaning again.) Pitch manipulation numerically by octaves is done modulo 8, and it becomes possible to deal with pitches as an ordered set of contiguous integers without having to abandon a widely accessible musical vocabulary.

To sum up (and go a bit further), here is a list of frequency sets, each being a subset of the set immediately above it.:

1. Continuous frequency: theoretical, actual, or analog:

0 (----------)

2. Audible frequency, discrete but perceived as continuous as it approaches or matches the resolution of our ears' discrete frequency sensing mechanisms:

> 30 Hz (----------) < 20 Khz

3. Quantized equal tempered scales: (Repeat all patterns from here on down over additional octaves by adding or subtracting multiples of 12)

Spacing:

(! ! ! ! ! ! ! ! ! ! ! !)

Conventional note names:

(c c# d d# e f f# g g# a a# b)

Interval from last note in semitones:

(1 1 1 1 1 1 1 1 1 1 1 1)

Interval from tonic in semitones:

(0 1 2 3 4 5 6 7 8 9 10 11)

4. Tonal subset of the equal tempered set:

As pattern or template for above:

```
         ( !  -  !  -  !  !  -  !  -  !  - !)
```

Conventional note names:

```
         ( c  -  d  -  e  f  -  g  -  a  - b)
```

Interval from last note in semitones:

```
         ( 1     2     2  1     2     2    2)
```

Offset from tonic in semitones:

```
         ( 0     2     4  5     7     9   11)
```

Condensed version of above as it would appear
in an array of scale degree intervals:

```
Note names                 (c d e f g a b )
Interval from last member  (0 2 2 1 2 2 2 1 )
Offset from tonic          (0 2 4 5 7 9 11)
Index within scale         (0 1 2 3 4 5 6 )
```

The common Gregorian modes can be generated from the above subsets of the equal tempered scale by rotating the starting index within the tonal set modulo 12, or by creating a longer array which repeats the above offset cycle adding successive multiples of 12 for successive octaves:

```
           (c-d-ef-g-a-bc-d-ef-g-a-b)
Major      (!-!-!!-!-!-!!)
Dorian       (!-!!-!-!-!!-!)
Phrygian       ( !-!-!-!!-!-!)
Lydian          (!-!-!-!!-!-!!)
Myxolydian        (!-!-!!-!-!!-!)
Aeolian             (!-!!-!-!!-!-!)
Locrian               (!!-!-!!-!-!-!)
```

Putting this in other terms usable with arrays:

Mode	Interval	Offset
Major	2 2 1 2 2 2 1	or 0 2 4 5 7 9 11 12
Dorian	2 1 2 2 2 1 2	0 2 3 5 7 9 10 12
Phrygian	1 2 2 2 1 2 2	0 1 3 5 7 8 10 12
Lydian	2 2 2 1 2 2 1	0 2 4 6 7 9 11 12
Myxolydian	2 2 1 2 2 1 2	0 2 4 5 7 9 10 12
Aeolian	2 1 2 2 1 2 2	0 2 3 5 7 8 10 12
Locrian	1 2 2 1 2 2 2	0 1 3 5 6 8 10 12

5. Triadic subsets of the tonal set (From here down use modulo 8 arithmetic.):

As pattern or template for the major scale:

```
Tonal set    (c d e f g a b c)
Names        (c - e - g - - c)
```

```
 Intervals    (0 - 2 - 2 - - 3)
 Offsets      (0 - 2 - 4 - - 7)

Chord quality is automatically tonally correct
when chords are described as rotations
within this ordered subset or array:

       c d e f g a b
      (! - ! - ! - - )  C (I)   major
      (- ! - ! - ! - )  D (ii)  minor
      (- - ! - ! - ! )  E (iii) minor
      (! - - ! - ! - )  F (IV)  major
      (- ! - - ! - ! )  G (V)   major
      (! - ! - - ! - )  A (vi)  minor
      (- ! - ! - - ! )  B (vii) diminished

 or to make the pattern more visible:
    cdefgabcdefgabc
    012345601234560

    !.!.!..!         I    major
     !.!.!..!        ii   minor
      !.!.!..!       iii  minor
       !.!.!..!      IV   major
        !.!.!..!     V    major
         !.!.!..!    iv   minor
          !.!.!..!   vii  diminished
```

At this point, when we make triads, we no longer have to keep track what kind of chord qualities they have (major, minor, diminished), or of the differing numbers of semitones in the internal intervals of these different tonal chords.

All triads consist of offsets of 0, 2, and 4 added to any number within the tonal pitch set (0 to 7), that number indicating the root of the chord. The triadic pattern moved as a template through the ordered tonal set automatically results in a triad of the proper chord quality when the tonal array in its turn is used as a template for indexing into the equal tempered scale array.

This conceptual organization makes it much easier to compute musical material by algorithm without abandoning the harmonic vocabulary in which non-computer music has most effectively reached us.

The above in less abstract and more useful form:

More computerishly put, this Set Theory of Musical Pitch includes these types of sets (Continua, Contigua, Collections, Chords (including Clusters), and Cycles):

#1. Continuum - Frequency: Not applicable to computers.

#2. Contiguum - Audible frequency. Limited by hardware for sound generation, by word size, and by the physiology of hearing. (I propose the term "contiguum" for sets of discrete contiguous elements):

#3. Contiguum - Quantized scale: An array containing values to output to oscillator hardware. These can be equally quantized (tempered, microtonal, or macrotonal scales), or much more bizarre.

#4. Collection - Tonal set: An array of indices into the equal tempered scale in #3 above as follows: 0,2,4,5,7,9,11...

#5. Chord - Triadic subset: Array of indices into #4 above (major = 0,2,4,7...).

Other musically useful arrays which can be expressed as subsets of the tonal collection:

#6. Collection - Pentatonic (0,1,2,4,5) (= C,D,E,G,A). This collection is a subset of the major scale which excludes the two single-semitone intervals and the tritone (an interval of 6 semi-tones, or 3 tones), leaving only intervals of a wholetone or larger.

#7. Chord - Fourth chords (0,1,2,4,5) (= C,D,G,A), which are wrap-arounds of the cycle of fifths. (I use the term "wrap-around" in from computer graphics to indicate a folding in modulo some finite range, in this case the octave.)

#8. Cycle - of fifths. The subset of the full cycle of fifths which falls within a single key signature, without adding sharps or flats, is (as offsets wrapped around into one octave: 3,0,5,2,6) (= F,C,G,D,A,E,B), is also useful. (These can be added to the index into the equal tempered scale array, for modulation to related keys in tonal pieces.)

```
#9. Cycles - Equal divisions of the octave:

        a. By 1/12 = equal tempered scale.
        b. By 1/6  = whole tone scale.
        c. By 1/4  = diminished seventh chord.
        d. By 1/3  = augmented chord.
        e. By 1/2  = the tritone (and octave).

#10. Cycle - Alternate major and minor thirds:

        c e g b d f# a c# e g# b d# f# ...

#11. Cycle - Dovetailed minor seventh chords (courtesy of Ron Everett in
Toronto). These consist of a repeating cycle of a major third followed
by two minor thirds:

        c e g b-flat d f a-flat c e-flat g-flat ...

#12. Composed - Those I'll leave to all of our imaginations.
```

Levels of Indirection in Musical Progression

The above is all good useful stuff for algorithmic composition, but conceptual mechanisms for referencing and assembling scaler, triadic, and other patterns are still insufficient for the composition of tonal progressions. For the creation of chord progressions, I have found it useful to conceive of them as series of levels of indirection (removal) from a destination chord (generally the tonic). These can also be conceived of as nested loops, concentric circles, or in a tree configuration.

As a tree-like structure, all branches connect at nodes where selection is done, such that we move in, layer by layer toward the single central tree trunk. The difference between the standard binary tree model and what I describe here is twofold.

First, the purpose of this tree is not finding something which has been stored, not for searching. Its purpose is to provide a means of structuring paths of movement. Music is process, progression. (Arrival in music may threaten to stop the music. Note how Bach elides cadences, so that a chord which is heard as a strong arrival, or ending, is set up to be simultaneously heard as the beginning of a new phrase.)

Second, the direction of movement is opposite to that of a tree search. Instead of starting at a central known place and selecting where to branch to next, we may find ourselves just anywhere up in the branches, in the structure of possibilities of a given moment, and we know we always want to move one level down toward the trunk. We don't care what branch we're currently in, but must know how many levels out we are from the trunk so we know by what means to select where to go in the next level down. We wish to do this in some aesthetically effective manner, somehow not too obvious or expected, not consistent or predictable, but not too shocking, awkward, or unpredictable either. There must be underlying feelings of a sense of direction and that where we've arrived, though unexpected, makes sense.

As an example, here is an algorithm for a simple chord sequence generator based on the idea of levels of indirection from a chord of harmonic resolution, using frequency subsets (expressed as arrays of indices into other arrays, as per the above):

Tree of Levels of Indirection from Tonic chord:

Arrays used include the equal tempered scale ("etscale"), our 7 tonal intervalic offsets from the tonic ("tonality"), and the 3 offsets to the tonic which define a chord ("triad").

```
4 - Leaves:
----------
iii chord (can go to other places besides the 2 chords on the next level
down, but let's start simply).

        REM play arpeggiated triad
        tonic = 3
        for i = 0 to 3
        note =etscale(tonality(tonic + triad(i)))
        next i

        REM select next chord by weighted probability
        REM (or use other means)
        which = random mod 25
        if which < 9 then tonic = 1
        else tonic = 6

3 - Twigs:
---------
I chord or vi chord (can go to either chord on next level down).

        (Repeat 2 steps above, with current values.)

2 - Branches:
------------
IV or ii chords (can go to either chord on next level down).

        (Repeat 2 steps above, with current values.)

1 - Boughs:
```

```
----------
V or vii chords (can go to either chord on next level down).

        (Repeat 2 steps above, with current values.)

0 - Trunk:
---------
I (tonic) or vi (tonic of relative minor key)

        REM keep going even though we got here.
        level = random mod 5
        goto level
```

In case it isn't utterly and completely obvious, the above is simply a statement of logic, and will have to be coded for whatever machine, language, and oscillator hardware you may use. (I have a more extensive version of this algorithm running on my Apple II and Mountain Hardware oscillator boards, as part of my PASCAL composing system, "AMO" (A Musical Offering). Rather than including a printout, though, I thought it would be clearer to state this idea in general form, so that it may be better understood, more widely tried out, played with, and adapted to different music systems.)

Though the 2 possible chords on each level of this tree have roots a third apart, I chose not to describe them as single arrays containing the union of the 2 triads, to select them by choosing a starting index of either 0 or 1 into each union array (vi-I, ii-IV, V-vii) because that would have limited the variability and generality of the algorithm.

The aesthetics of this algorithm, like many others which are possible, its personalizability and the nature of its expression, lie in the choice of which chords (clusters, modes, etc.) may be moved between, and in what order, and in the design of the method of decision making (in this case the simple weighting of the probabilities). I suggest starting, if you do decide to try running it, with the chords I have selected, and with 50-50 probabilities on all levels, before trying to vary it. (To make sure this progression algorithm really worked musically, I wanted to first test it within a harmonic style derived from J.S. Bach, which accounts for the chords used and their positions.)

As it stands above, variables available to play with include probabilities, the number of levels used, and the chords or tone clusters, modes or scales which are entered in the arrays. Among the variables not dealt with in the above are tempo, timbre, stereo location, density (number of simultaneous voices), range in octaves, and envelope parameters. Elaborative processes which could be added include the introduction of other shapes of arpeggios, the generation of melodic lines from the chords, instead of arpeggios, by use of leaps and passing tones, or the substitution of patterns (trills, tremolos, and other ornaments, or of motives or melodies) for notes.

Generalizing to Other Musical Dimensions

The structure applied above to chord progression can be used as a model for software which can generate routings through other areas of our frequency map as well. This can be done within any single level (for example, among scales instead of among chords), or to structure movement from one level of our set map (one area of our sonic subset theory) to another, that is to be used to redefine the actual working premises of our musical reality during a piece.

The same principles of organization can also be applied to other dimensions of music. Rhythm furnishes a good example, as it can easily be viewed as hierarchical (as a branching tree). The architecture of time can be hierarchically described, from 32 to 16 measure units, to 8 bar phrases, and on down to beat to beat rhythms. Progressions of harmony and rhythm merge in the concept of "harmonic rhythm" (the number of beats between regularly spaced chord changes), which can also be manipulated algorithmicly. Rhythmic meter can be viewed as a branching tree, in which each beat-level can be subdivided in a variety of selectable ways. (A quarter note can be divided into 2 eighth notes, a 3 eighth note triplet, 4 sixteenth notes, etc.).

What, Why, and Whither Algorithmic Music?

Algorithms may be viewed as general compositional processes, as are canon and fugue, or as unique musical compositions, or as falling someplace between those 2 extremes. Depending on their degree of interactivity (whether variables are read in from knobs, switches, keyboards, or other devices during a program run, or whether they are generated or stored within the software), such algorithms may be placed anywhere along the axis between "intelligent instruments" and "automated composition."

Algorithms may become a dominant musical form in the future for a number of reasons. They can be highly user interactive via realtime access to variables (anywhere from audio game to virtuoso's instrument). They are naturals for the powerful musical instruments of the future which will permit maximal musical expression with minimal input, by the use of an increasingly small, select, and aesthetically powerful group of variables. They will be attractive to use by virtue of their overcoming the mechanical limitations of traditional instruments whereby the ratio of the number of notes played by the person to the number of notes played by the instrument is rarely better than one-to-one. Musical pattern generation, manipulation, and editing programs will greatly facilitate composition, as well as having wide application in music education.

Algorithmic music may turn out to be most desirable (so much so that economics may ensure its proliferation as a new musical common practise) because algorithms involving relatively small numbers of relatively powerful variables are extremely economical in terms of the amount of storage space necessary to describe a piece of music. In other words, the amount of stored (or telecommunicated) data necessary "per thousand" of musical notes played is small. Because a small number of variables can control the nature of a musical fabric, there is also a great potential for musical response to other technologically interfacable phenomena, such as temporal visual compositions.

The superset of high level variables can include global musical dimensions such as ranges, types of interpolation or transition (most easily done by selecting one of a group of a precomposed tables), rates (of acceleration, crescendo, or harmonic change). It may also include the selection or conditional use of such musical transformations as are described in my paper on "musical manipulations".

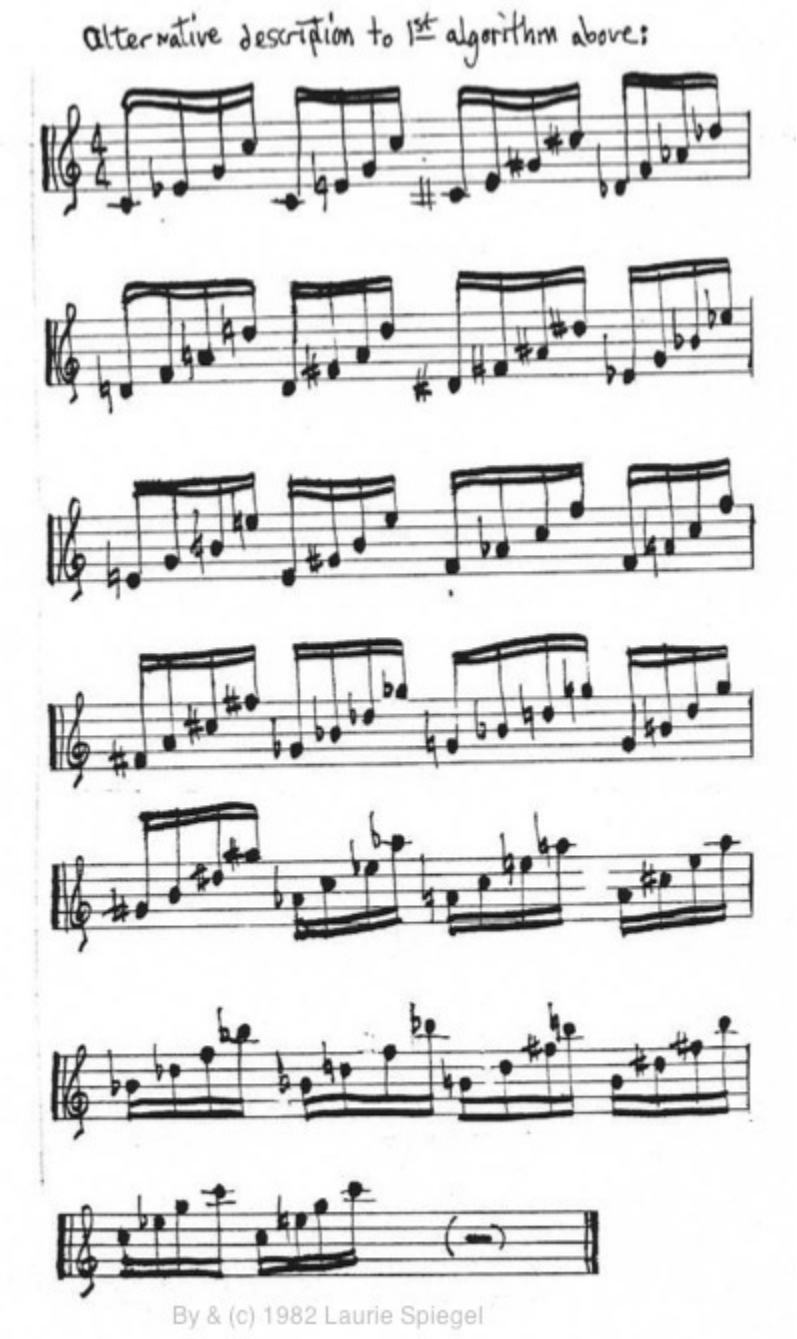
The algorithmic selection and variation of musical decision making processes themselves is an obvious and interesting recursion. Other important parameters, processes, and principles will be derived from theories of perception, cognition, games, information, and communication, and may include density, level of "contrast" among sonic components, percentage of redundancy, degree of continuity, type of organizational structure, and others.

Algorithmic composition is a concept applicable to other arts besides music, one example being the "game of life" in computer graphics. Algorithmic interfaces of audible and visual musics will become increasingly meaningful as variables become more powerful and more cognitively oriented (above the level of any single sensory modality).

The ultimate algorithmic artworks may hope to describe, by rules based on understanding, all the characteristics of stimuli which are most meaningful to our consciousness, externally embodying the mirror reflection of the structure of that consciousness and its mechanisms of finding such meaning. Algorithms may eventually simulate the transformations which sound and image undergo in our imaginations, which have been so difficult to capture by conventional methods of musical and artistic expression.

The basic tools above are only a most simplistic beginning, designed to facilitate broader participation in all that waits to be explored.

Laurie Spiegel
New York City
August 1982



Laurie Spiegel's Obsolete Systems page | *Laurie Spiegel's writings page* | *Laurie Spiegel's home page*