Gretchen C. Foley* **and Charles A. Cusack**†
*School of Music
University of Nebraska-Lincoln
Lincoln, Nebraska 68588-0100 USA
gfoley2@unl.edu
†Westshore Design, LLC
494 Lincoln Avenue, Suite 10
Holland, Michigan 49423 USA
ferzle@yahoo.com

Computer-Assisted Analysis of Music in George Perle's System of Twelve-Tone Tonality

Music theorists face many challenges when analyzing music written according to George Perle's compositional theory of *twelve-tone tonality,* a system based on inversional symmetry. Analysts focusing on Mr. Perle's music seek relationships among collections of notes in an effort to discover connections within and across a specific work. It is often possible to rely solely on the analytical tools of twelve-tone tonality when working with Mr. Perle's music, but in many instances other tools have proven rather useful, such as can be found in pitch-class set theory and transformational theory. The interested reader may refer to a number of excellent resources to acquire a working knowledge of these theories (Forte 1973; Rahn 1980; Lewin 1987; Straus 2005). However, in some cases even these theories do not assist the analyst in generating meaningful results from the pitch material in a twelve-tone tonal composition.

Because Mr. Perle's music is non-tonal, it lends itself particularly well to computer-assisted mathematical anlysis. We have developed a simple computer application, *Twelve-Tone Tonality Reverse Engineer,* or *T3RevEng,* that helps determine the organization and context of the material, based on the input of pitch classes expressed as integers. In all tests thus far, T3RevEng has proven to be an essential tool for the Perlean analyst owing to its comprehensive, efficient, and accurate manipulation of data.

The Fundamentals of Twelve-Tone Tonality

The foundation of twelve-tone tonality is provided by the *interval cycle,* defined as a series of pitch classes based on a single recurring interval, measured in half steps. Mr. Perle combines two inversionally related interval cycles to form what he calls a *cyclic set,* by placing the members of the cycles in alternation. Figure 1 illustrates a cyclic set based on the interval 1 cycle, with the ascending cycle in white noteheads, and the descending cycle in black noteheads. In the following figures, integers 0 to 11 represent the twelve pitch classes, with C = 0, C-sharp/D-flat = 1, D = 2, and so on. The integers 10 and 11, representing B-flat and B respectively, are replaced by the letters "t" and "e" to retain a single digit for each pitch class.

In any segment of the cyclic set comprising three adjacent pitch classes, the central note is referred to as the *axis note,* flanked on either side by *neighbor notes.* In any such segment, the axis note forms a pair of sums with the neighbor notes. These sums repeat with each three-note segment throughout the cyclic set and are called *tonic sums.* The tonic sums provide the cyclic set with its name; hence, the cyclic set in Figure 1 is 0,1. The sums 0 and 1 recur by adding pairs of successive integers: 0 + 0 = 0, 0 + 1 = 1, 1 + e = 0 (modulo 12), e + 2 = 1 (modulo 12), and so on.

The primary pre-compositional structure in twelve-tone tonality is the *array.* The array is constructed from any two vertically aligned cyclic sets, which provide it with its name. In Figure 2a, two cyclic sets built from interval 1 cycles combine to form the array 0,1/4,5. Arrays can also combine cyclic sets of different cyclic intervals. In Figure 2b, the array 0,4/4,5 contains cyclic sets constructed from interval 4 and interval 1 cycles.

A single interval 4 cycle does not contain the aggregate, that is, the collection of all twelve pitch classes in the modulo 12 universe. Instead, the aggregate partitions into four different interval 4 cycles, each containing only three pitch classes: (0-4-8), (1-5-9), (2-6-t), and (3-7-e). To form a cyclic set, either the same or different partitions may be

Computer Music Journal, 30:3, pp. 53–66, Fall 2006
© 2006 Massachusetts Institute of Technology.

Figure 1. Cyclic set 0,1.

Figure 2. Cyclic sets combine to form arrays. (a) Cyclic sets of the same interval form array 0,1/4,5. (b) Cyclic sets of different intervals combine to form array 0,4/4,5. (c) Three different alignments of cyclic sets from array 0,4/4,5.

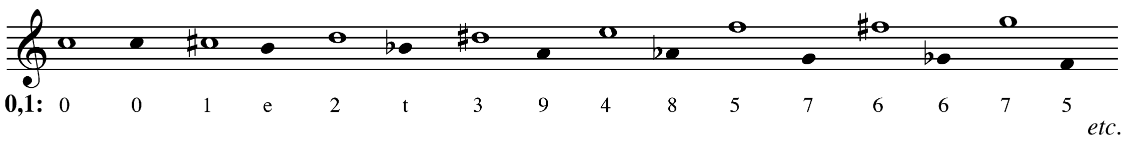


Figure 1

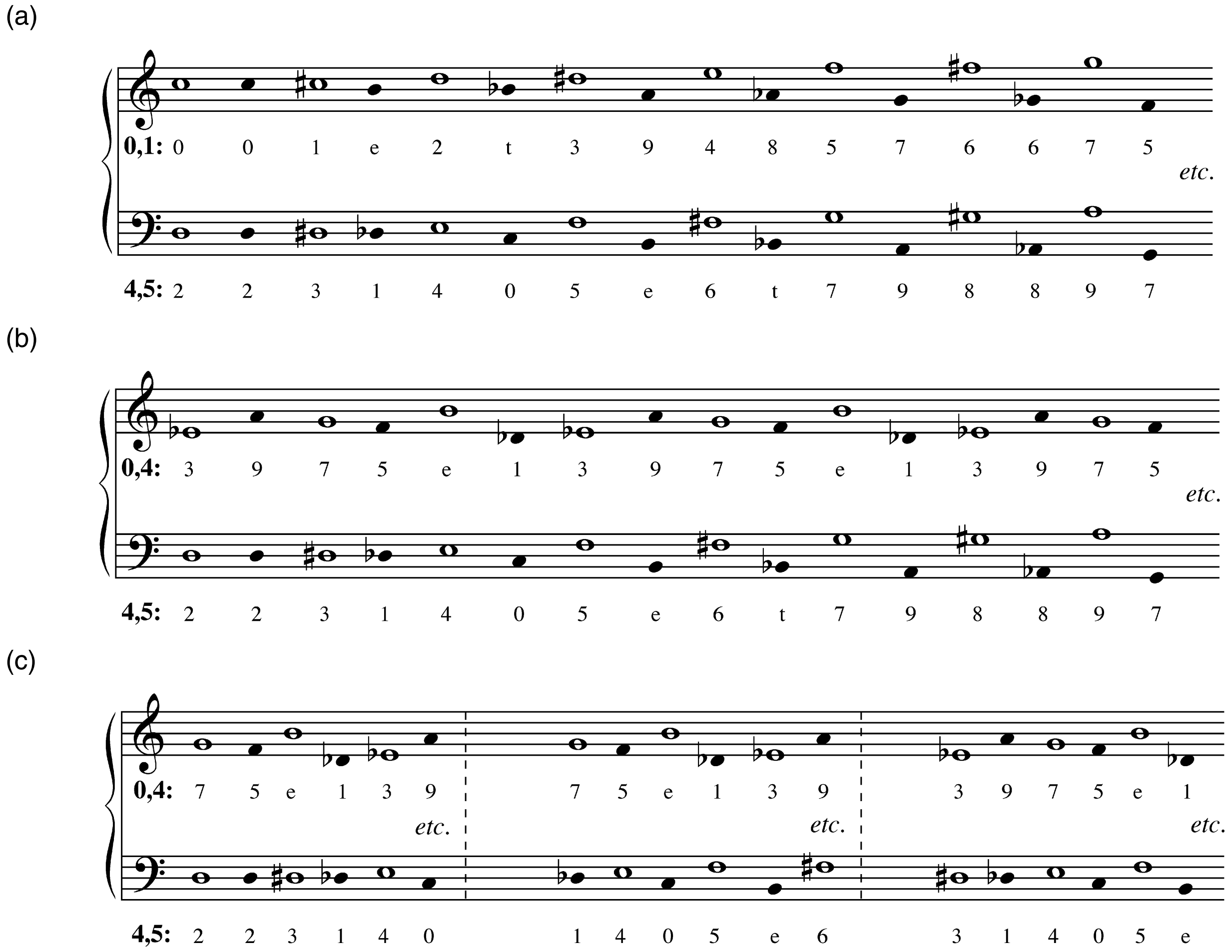
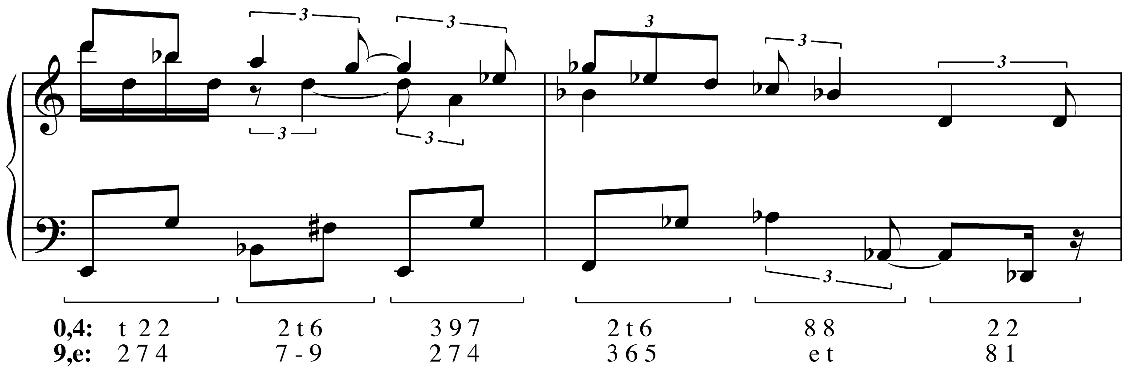

(a)
0,1: 0 0 1 e 2 t 3 9 4 8 5 7 6 6 7 5
etc.
4,5: 2 2 3 1 4 0 5 e 6 t 7 9 8 8 9 7
(b)
0,4: 3 9 7 5 e 1 3 9 7 5 e 1 3 9 7 5
etc.
4,5: 2 2 3 1 4 0 5 e 6 t 7 9 8 8 9 7
(c)
0,4: 7 5 e 1 3 9
etc.
7 5 e 1 3 9
etc.
3 9 7 5 e 1
etc.
4,5: 2 2 3 1 4 0
1 4 0 5 e 6
3 1 4 0 5 e

Figure 2

combined. In the array given in Figure 2b, the cyclic set 0,4 comprises two different partitions of inversionally related interval 4 cycles: (3-7-e) and (1-5-9). Moreover, the two cyclic sets of an array are not fixed in relation to one another; the composer may shift them horizontally in the manner of a slide rule to create different vertical alignments between them, as seen in Figure 2c.

To generate melodic and harmonic material at the musical surface, Mr. Perle fragments arrays into units ranging from trichords to hexachords. The main unit is the *axis-dyad chord,* a hexachord comprising two linear trichords segmented from the array's cyclic sets. From another perspective, the axis-dyad chord contains three vertical dyads, with the axis dyad in the center surrounded by two

Figure 3. Bracketed segmentations based on array 0,4/9,e. (Mm.1–2 of Etude No. 4, from Six Etudes for Piano *by George Perle. Copyright © 1978 [renewed] by G. Schirmer, Inc. [ASCAP]. International copyright secured. All rights reserved. Used by permission.)*



0,4: t 2 2 | 2 t 6 | 3 9 7 | 2 t 6 | 8 8 | 2 2
9,e: 2 7 4 | 7 - 9 | 2 7 4 | 3 6 5 | e t | 8 1

neighbor dyads. Another significant unit, the *sum tetrachord,* comprises two dyadic segments from the cyclic sets. As its name indicates, the sum tetrachord contains two of the four tonic sums of the array. Figure 3 shows an excerpt with segmentations drawn from various alignments of the array 0,4/9,e. Although the segmentations given below the staff suggest an ordered two-voice texture, the actual notes do not appear at the musical surface in the same ordered configuration. Rather, the composer may realize the notes of the segmentation freely, in an infinite variety of settings.

Furthermore, Mr. Perle interprets multiple occurrences of a pitch class within a segment as independent entities. Thus, a hexachord in Mr. Perle's system may actually contain fewer than six distinct pitch classes. The first axis-dyad chord in Figure 3, for example, comprises six notes, yet contains only four different pitch classes: {2, 4, 7, t}. A number of resources are available to readers seeking further information on the composer's theory of twelve-tone tonality (Perle 1977, 1996; Carrabré 1993; Headlam 1995; Rosenhaus 1995; Foley 1999).

Segmentation Considerations

In studying any of Mr. Perle's compositions in twelve-tone tonality, the analyst typically begins by establishing the interval cycles and arrays of the music. Using an axis-dyad chord from Figure 3, Figure 4a clarifies how the requisite information is obtained. The cyclic set is named by the tonic sums created by adjacent pairs of notes, and the originating interval cycles are found in the difference between the corresponding neighbor notes. In Figure 4a, the adjacent pairs of notes identify the upper cyclic set as 0,4 (because 3 + 9 = 0 and 9 + 7 = 4, modulo 12), while the difference between the neighbor notes reveals the underlying interval 4 cycle (because 7 – 3 = 4). Similarly, the sums created by the adjacent pitches in the lower cyclic set provide its name (9,e), and the difference between the neighbor notes determines the underlying interval 2 cycle. Thus, the axis-dyad chord contains within its structure all the information necessary to determine its fundamental array, based on the positioning of the chord's component pitch classes. It is possible, however, to rearrange these same pitch classes {2, 3, 4, 7, 7, 9} to suggest other arrays based on cycles of interval 4 and interval 2, as illustrated in Figures 4b, 4c, and 4d.

Furthermore, the number of array possibilities increases dramatically if the interval cycles of the array have not yet been established. For example, the same collection of pitch classes from Figure 4 can be arranged to form 360 different arrays based on various combinations of other interval cycles.

Segments that contain only four or five pitch classes pose substantial challenges for the analyst, however, because it is not possible to arrange the notes in such a way as to display all four tonic sums and both interval cycles, and thereby unequivocally reveal the underlying array. To illustrate, the pitch classes of the pentachordal segment in Figure 5 have been arranged to show either (a) both interval cycles and two tonic sums, or (b) one interval cycle and three tonic sums. The pitch classes of the tetra-

Figure 4. Axis-dyad chords reveal both interval cycles and all four tonic sums of array.

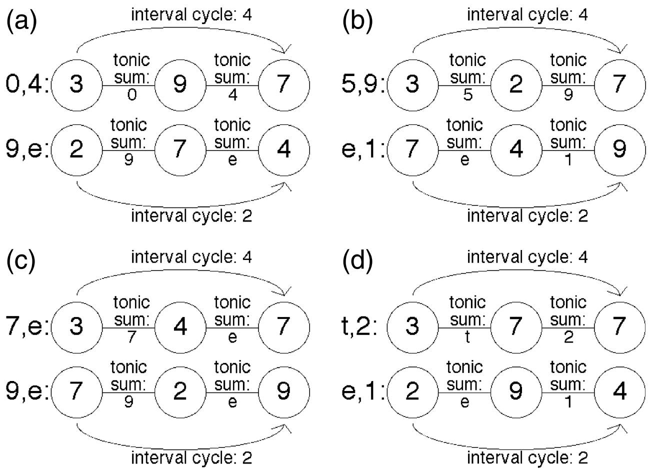
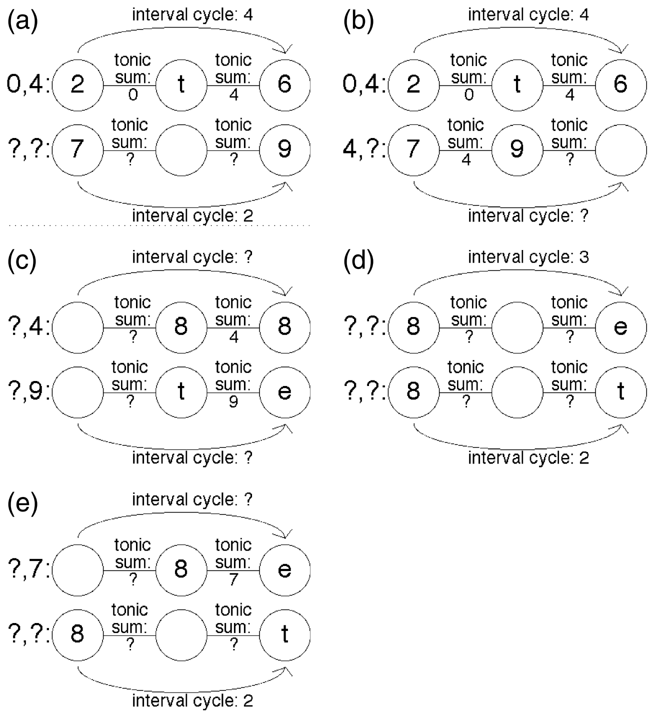

(a)
interval cycle: 4
0,4: 3 tonic sum: 0 9 tonic sum: 4 7
9,e: 2 tonic sum: 9 7 tonic sum: e 4
interval cycle: 2
(b)
interval cycle: 4
5,9: 3 tonic sum: 5 2 tonic sum: 9 7
e,1: 7 tonic sum: e 4 tonic sum: 1 9
interval cycle: 2
(c)
interval cycle: 4
7,e: 3 tonic sum: 7 4 tonic sum: e 7
9,e: 7 tonic sum: 9 2 tonic sum: e 9
interval cycle: 2
(d)
interval cycle: 4
t,2: 3 tonic sum: t 7 tonic sum: 2 7
e,1: 2 tonic sum: e 9 tonic sum: 1 4
interval cycle: 2

Figure 5. Pentachordal and tetrachordal segmentations convey varying amounts of array information.


(a)
interval cycle: 4
0,4: 2 tonic sum: 0 t tonic sum: 4 6
?,?: 7 tonic sum: ? tonic sum: ? 9
interval cycle: 2
(b)
interval cycle: 4
0,4: 2 tonic sum: 0 t tonic sum: 4 6
4,?: 7 tonic sum: 4 9 tonic sum: ?
interval cycle: ?
(c)
interval cycle: ?
?,4: tonic sum: ? 8 tonic sum: 4 8
?,9: tonic sum: ? t tonic sum: 9 e
interval cycle: ?
(d)
interval cycle: 3
?,?: 8 tonic sum: ? tonic sum: ? e
?,?: 8 tonic sum: ? tonic sum: ? t
interval cycle: 2
(e)
interval cycle: ?
?,7: tonic sum: ? 8 tonic sum: 7 e
?,?: 8 tonic sum: ? tonic sum: ? t
interval cycle: 2

chordal segment have been arranged to show either (c) two tonic sums, (d) two interval cycles, or (e) one tonic sum and one interval cycle. Only the axis-dyad chord embodies the structure capable of showing all four tonic sums and both interval cycles.

Arranging a segment of six notes into an axis-dyad chord is equivalent to giving a linear ordering of the six notes, where the first three notes are the upper trichord and the last three notes are lower trichord. Thus, the number of arrangements of six notes into an axis-dyad chord is the number of permutations on six numbers, namely 6! = 720 (recall that n! is read as "n factorial," and it is defined by $n! = n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1$), assuming the pitch classes are all unique. Of course, this number is smaller if some of the pitch classes are repeated, as we saw with the pitch classes {2, 3, 4, 7, 7, 9} above. In general, the number of ways to arrange k distinct pitch classes (and $6-k$ unknowns) into an incomplete axis-dyad chord is $k! \times C(6,k) = 6!/(6-k)!$ because we need to choose the k spots to place the notes (C(6,k) ways), and then we must consider every possible ordering of the notes in these spots (k! ways). Here, C(n,k) is the number of ways of selecting k elements from a set of n elements and is defined by $C(n,k) = n!/k!(n-k)!$. As we will see, not all of these arrangements must be considered in order to generate the list of possible arrays, because different arrangements can yield the same array.

Every arrangement of a chord has a potentially different array. In fact, for some segments of six notes, all 720 arrangements yield different arrays. On average, a segment of six notes has 237 different possible arrays, as demonstrated later in Table 3.

The situation is slightly more complicated for segments of three to five notes. To compute the possible arrays in this case requires enumerating all possible arrangements of the given notes into an axis-dyad chord, and filling in the "missing" notes with every possible pitch class. Table 1 gives the list of all possible configurations of axis-dyad chords given a segment of k = 3, 4, 5, or 6 notes. We use the letters *a, b,* and *c* to identify the first, second, and third spots in the upper trichord, and *d, e,* and *f* to identify the first, second, and third spots in the lower trichord. This allows us to use shorthand notation—for example, [ac|e] can denote the segment configuration:

```
a-c
-e-
```

The *type* of the configuration denotes the number of known notes in each trichord, so the configuration [ac|e] is of type 2 – 1, because two notes in the

Table 1. Possible Segment Configurations Consisting of *k* = 3, 4, 5, or 6 Notes

k	*Type*	*Segment Configurations*								
3	3-0, 0-3	abc ---	--- def							
3	2-1	ab- d--	ab- -e-	ab- --f	a-c d--	a-c -e-	a-c --f	-bc d--	-bc -e-	-bc --f
3	1-2	a-- de-	a-- d-f	a-- -ef	-b- de-	-b- d-f	-b- -ef	--c de-	--c d-f	--c -ef
4	2-2	ab- de-	ab- d-f	ab- -ef	a-c de-	a-c d-f	a-c -ef	-bc de-	-bc d-f	-bc -ef
4	3-1, 1-3	abc d--	abc -e-	abc --f	a-- def	-b- def	--c def			
5	3-2, 2-3	abc de-	abc d-f	abc -ef	ab- def	a-c def	-bc def			
6	3-3	abc def								

Table 2. Arrangements of {1, 3, 5, e} Yielding Array 4,8/2,7

Configuration	*Missing Notes*	*Chord*
abc d--	3, 4	**1 3 5** **e** 3 4
abc -e-	3, 8	**1 3 5** 3 **e** 8
abc --f	6, 8	**1 3 5** 6 8 **e**

upper trichord and one note in the lower trichord are known.

The configurations shaded in gray are actually equivalent to others in the sense that they generate the same set of possible arrays. In fact, it is easy to see that when only one note in a given trichord is known, all of the 144 possible cyclic sets are possible for that trichord. In other words, the "lone note" creates no constraints. This means that, for instance, the configurations [abc|d], [abc|e], and [abc|f] are all equivalent. Because many of the possible configurations for two notes involve a lone pitch in both trichords, all arrays are possible, which is why we do not include them in our discussion. Table 2 shows how three different 3-1 configurations of the segment {1, 3, 5, e} can yield the same array.

The Need for a New Tool

Table 3 gives information about possible arrays for segments of $k = 3, 4, 5$, or 6 notes. The number of unique segments of k notes is $N(k) = C(12 + k - 1, k)$, and an upper bound on the number of arrays to check, $NA(k)$, is $k! \times 12^{(6-k)} \times C(6,k)$. As mentioned above, the actual number is less than this for $k = 3$ and 4, since not all of the $C(6,k)$ configurations need be considered. We use $NA_{avg}(k)$, $NA_{max}(k)$, $NA_{min}(k)$, to denote the average, maximum, and minimum number of arrays that a chord with k notes actually has. The minimum, which clearly occurs if all of the notes are the same, can be thought of as the best case for the analyst. For example, no matter what segment of four notes one is given, we know that at least 1,376 arrays are possible.

As is often the case, the definition of "average" can be misleading. The average segment in our analysis has many duplicate notes, because every possible multi-set of k notes is considered, whereas the average segment in an actual composition is more likely to have a small number of duplications. However, duplication of notes generally reduces the possible arrays, so the number of possible arrays for a set of notes from an actual composition is generally higher than the overall average. In other words, when analyzing a composition, it is likely that the number of possible arrays will be larger than the av-

Table 3. Possible Arrays for Sets of Chords with k = 3, 4, 5, or 6 Notes

				Maximum		*Minimum*	
k	$N(k)$	$NA(k)$	$NA_{avg}(k)$	$NA_{max}(k)$	*sample*	$NA_{min}(k)$	*sample*
3	364	82,944	≈17,241	19,215	0,1,3	8,636	0,0,0
4	1,365	38,016	≈7,887	11,454	0,1,3,5	1,376	0,0,0,0
5	4,368	8,640	≈2,030	3,968	0,1,2,5,9	67	0,0,0,0,0
6	12,376	720	≈237	720	0,1,2,3,4,7	1	0,0,0,0,0,0

erage we computed, so in some sense the average can be viewed as a better-than-usual case scenario.

The results of Table 3 were obtained by an exhaustive search of all possible segments of three to six notes. For six notes, we computed all possible arrays for every possible segment of six notes by considering all 720 arrangements of the segment into an axis-dyad chord. When $k < 6$, we must exhaust every possible segment of k notes, filling in the "missing" $(6 - k)$ notes with every possible combination of pitch classes (clearly, there are $12^{(6-k)}$ possibilities) and consider every ordering of the given k notes for every non-equivalent configuration from Table 1. The number of steps required to compute the numbers $NA_{avg}(k)$, $NA_{max}(k)$, $NA_{min}(k)$, is on the order of $\mathrm{N}(k) \times \mathrm{NA}(k)$, which is exponential in k. However, because $k \leq 6$, the number of steps involved is actually not that large, assuming a computer is available. In fact, these numbers were all computed in a matter of minutes on a Mobile AMD Athlon 64 Processor 3200+ with a clock speed of 1.60 GHz and with 512 MB of RAM. As an example of how the computation was performed, we will describe in more detail the process of determining these numbers for tetrachordal segments.

To determine the possible arrays for a segment of four notes requires checking eleven possible segment configurations, determined by where the two "missing" notes are in the chord. The careful reader will observe that there are actually 15 configurations, not 11. However, recall that each of the 3-1 (and 1-3) configurations generates the same possible arrays, so one must only check one of each of these configurations, reducing the number from 15 to 11. For each of the two missing notes, there are twelve possible pitch classes from which to choose, and 4! = 24 arrangements of the 4 given notes. The total number of possible configurations is thus $11 \times 12^2 \times 24 = 38{,}016$. For each of the 1,365 unique segments of 4 notes, we must check all of these configurations and enumerate the arrays. The maximum number of possible arrays for a segment with 4 notes is 11,454, with an average of 7,887.

The analysis for segments with three or five notes is similar to that of four notes. Notice that because there are only $12^4 = 20{,}736$ possible arrays, segments of three notes are not very helpful in narrowing down the array used to create the chord, because on average 17,241 (more than 83%) of the arrays are possible.

Given these results, it is clear that determining the possible arrays by hand for even a single chord can be prohibitive. Further, how does one determine the arrays for a set of chords? The obvious method is to determine the set of possible arrays for each chord, and then find the intersection of all of these sets. Again, this is practically impossible to do by hand. It was in light of these facts that we developed the T3RevEng, a software analysis tool that makes these and other tasks trivial to perform in a matter of seconds.

The Tool Described

Other software tools exist for Perlean analysis. In 1994, James H. Carr and Charles Porter, former protégés of George Perle, developed a Macintosh application named "12-tt 2.0." This software lists arrays when the interval system and the sums are specified (Carr 1995). In describing the program to these authors, Mr. Carr referred to it as an "array encyclopedia," and stated that Mr. Perle used it for many years. More recently, another former student of

Mr. Perle's, Dave Headlam, created a series of programs for a graduate seminar he gave at the Eastman School of Music in 2005. The summary he provided for the authors through personal correspondence is as follows.

When provided with an input of from four to six pitch classes, these programs identify all possible array names, but only for one pair of interval cycles at a time, as specified by the user. In addition, one of Mr. Headlam's students at the Eastman School, Christopher Winders, wrote a program that allows the user to slide the cyclic sets of an array in relation to one another to produce all possible vertical alignments, and some information about that array's mode and key (more abstract entities in twelve-tone tonality). Our program, T3RevEng, goes further than its predecessors in that it allows the user to determine all possible arrays, interval cycles, and tonic sums of a passage composed in twelve-tone tonality. Specifically, it can show every arrangement of a segment of six pitches into an axis dyad chord based on a given array name or interval cycles; draw sum squares, difference squares, and sum pentagrams; give all array names compatible with a segment of four pitches and a pair of interval cycles; and compute all array names and corresponding chord configurations compatible with any set of segments, each consisting of three to six notes. We will now briefly describe some of the tasks that T3RevEng can perform.

Axis-Dyad Chord Creation

Given a segment of six notes and either the desired array or the interval cycles of the desired array, T3RevEng determines all possible axis-dyad chords. For instance, if the user enters the notes {1, 2, 3, 4, 5, 6} and interval cycles 1,5 into the AD Chord (cycles) Task, the program will produce the following output:

```
There are 6 possibilities:
6,7: 243 7,8: 253 5,6: 324 8,9: 354
6,e: 156 5,t: 146 6,e: 156 3,8: 126

6,7: 425 7,8: 435
4,9: 136 3,8: 126
```

Note that all of the arrays have interval 1 and interval 5 cycles, and these six arrangements are the only possibilities in this case. If the user enters notes {2, 3, 5, 6, 7, 9} and array t,1/7,e into the AD Chord (array name) Task, the program will produce the following:

```
There is 1 possibility:
t,1: 376
7,e: 529
```

The algorithm used for both of these is the brute-force exhaustive search algorithm. That is, in each case, all 720 arrangements of the notes are considered, and the array or interval cycles are computed and matched against the input. Those that produce the same array or interval cycles are produced as output. At first glance, it seems that this approach is too simplistic, and a more elegant and efficient algorithm could be implemented. For instance, if the input is {1, 2, 3, 4, 5, 6}, and the array is 0,0/0,0, it is clear that no arrangement will work, and a check that no two of the notes add to 0 will reveal this.

Should we implement a cleverer algorithm that more quickly determines those arrangements that will work? Surprisingly, we should not for several reasons: the naïve algorithm was trivial to implement, is absolutely correct, and, because the number of arrangements is fixed at 720, it is extremely fast. This is an instance where the motto "work harder, not smarter" applies.

Squares and Pentagrams

As we have seen, tetrachordal and pentachordal segments do not contain enough information to determine the underlying array. However, different parts of the array can be determined if we make assumptions about the configuration of the chord. This is where T3RevEng's sum squares, difference squares, and sum pentagrams are useful. Figure 6 shows both a sum square and difference square for the segment {1, 2, 3, 4}, and Figure 7 shows a sum pentagram for segment {1, 2, 3, 4, 5}. The nodes contain the notes of the segment, and the edges are labeled with either the sum of the adjacent notes (in the sum square and sum pentagram), or both possible differ-

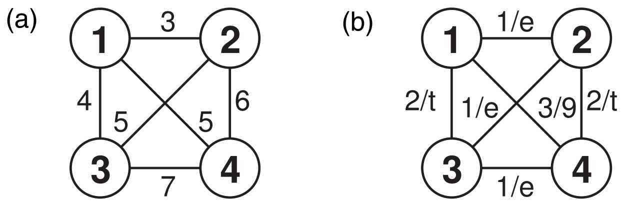


Figure 6. (a) Sum square and (b) difference square for segment {1, 2, 3, 4}.

ences between the adjacent notes (in the difference square), with all mathematics performed modulo 12. T3RevEng was used to generate the figures of the sum squares, difference squares, and sum pentagrams in this article.

The sum square is useful if the analyst assumes the four notes in the tetrachordal segment belong to one of the 2-2 configurations in which the notes in each trichord are adjacent (e.g., [ab|de], [ab|ef], etc.), because each partition of the edges of the sum square will then reveal one of the tonic sums for each trichord. Thus, from Figure 6a, we can see that for segment {1, 2, 3, 4}, the possible tonic sums pairs for each trichord are 3/7, 4/6, and 5/5. Note that these notes can be arranged to form array 3,2/7,4, for instance, but not 3,7/2,4, because the tonic sums 3 and 7 do not belong to the same cyclic set in the array.

If the cycle intervals are also known, the list of possible arrays is 24, because for each of the three partitions, there are eight ways of distributing the two known tonic sums in the array such that one occurs in each trichord. Given a tetrachord and the cycle intervals, the Sum Square Task of T3RevEng will output these possibilities. For instance, if we enter the segment {1, 2, 3, 4} and cycle intervals 3,5, the output will be

```
sum pair 3/7: 3,6 3,6 0,3 0,3 7,t 4,7 7,t 4,7
              7,0 2,7 7,0 2,7 3,8 3,8 t,3 t,3

sum pair 4/6: 4,7 4,7 1,4 1,4 6,9 3,6 6,9 3,6
              6,e 1,6 6,e 1,6 4,9 4,9 e,4 e,4

sum pair 5/5: 5,8 5,8 2,5 2,5 5,8 2,5 5,8 2,5
              5,t 0,5 5,t 0,5 5,t 5,t 0,5 0,5
```

A difference square can be used when the assumption is that tetrachordal segment came from the [ac|df] configuration. This particular arrangement of the pitches will reveal the various possibilities for

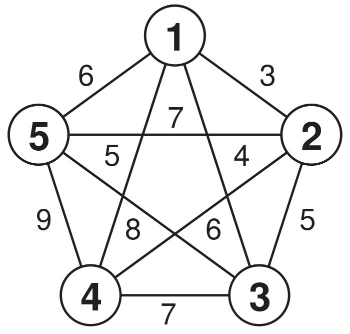


Figure 7. Sum pentagram for segment {1, 2, 3, 4, 5}.

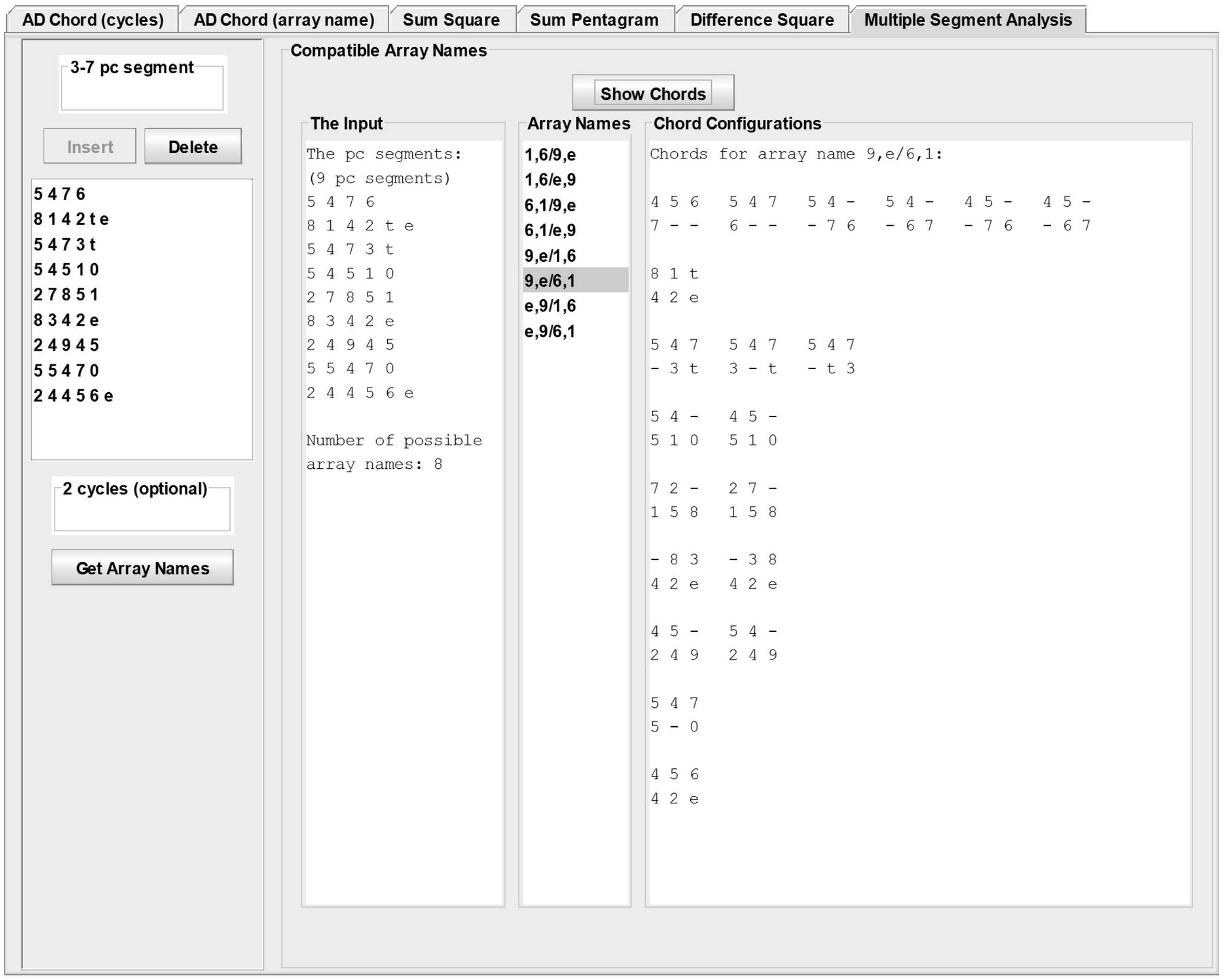
the interval cycles underlying the array. If the analyst suspects that the tetrachordal segment represents a neighbor chord rather than a sum tetrachord, the difference square will provide the configuration that corresponds to the prevailing interval cycles.

Given a pentachordal segment, we have already seen that we can either determine both interval cycles and two tonic sums, or one interval cycle and three tonic sums. The former is the case for configurations in which two notes are non-adjacent ([abc|df] and [ac|def]), and the latter is the case for the remaining configurations, and is also when the sum pentagram is useful. The development of the sum and difference squares and sum pentagrams led to the strongest utility of the software, the Multiple Segment Analysis task, which we describe next.

Multiple Segment Analysis

The ability to consider multiple segments simultaneously is crucial to analyzing compositions in twelve-tone tonality, because chords function as related entities within a passage, not as isolated elements. This is where T3RevEng really shines. Given a set of segments, each consisting of anywhere between three and six notes, the Multiple Segment Analysis Task of T3RevEng computes all arrays that are compatible with all of the segments. An array can then be selected, and it will show all of the non-equivalent configurations of each segment that conform to that array. In addition, a set of cyclic intervals can be entered to restrict the arrays to those whose interval cycles match. Figure 8 gives

Figure 8. Screenshot of T3RevEng Multiple Segment Analysis.



```
5 4 7 6
8 1 4 2 t e
5 4 7 3 t
5 4 5 1 0
2 7 8 5 1
8 3 4 2 e
2 4 9 4 5
5 5 4 7 0
2 4 4 5 6 e
```

The Input

```
The pc segments:
(9 pc segments)
5 4 7 6
8 1 4 2 t e
5 4 7 3 t
5 4 5 1 0
2 7 8 5 1
8 3 4 2 e
2 4 9 4 5
5 5 4 7 0
2 4 4 5 6 e

Number of possible
array names: 8
```

Array Names

```
1,6/9,e
1,6/e,9
6,1/9,e
6,1/e,9
9,e/1,6
9,e/6,1
e,9/1,6
e,9/6,1
```

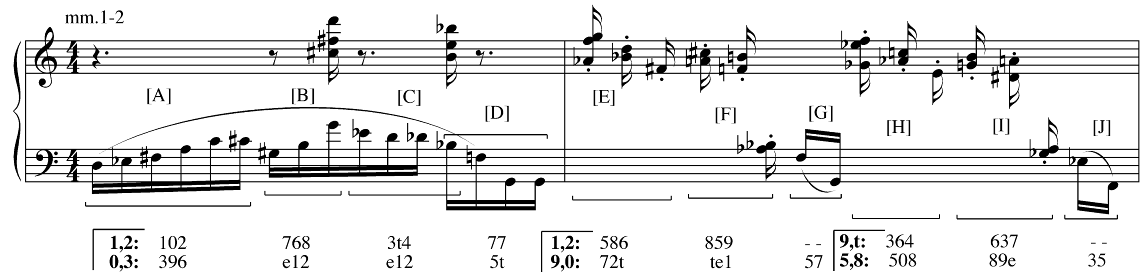
Chord Configurations

```
Chords for array name 9,e/6,1:

4 5 6   5 4 7   5 4 -   5 4 -   4 5 -   4 5 -
7 - -   6 - -   - 7 6   - 6 7   - 7 6   - 6 7


8 1 t
4 2 e

5 4 7   5 4 7   5 4 7
- 3 t   3 - t   - t 3

5 4 -   4 5 -
5 1 0   5 1 0

7 2 -   2 7 -
1 5 8   1 5 8

- 8 3   - 3 8
4 2 e   4 2 e

4 5 -   5 4 -
2 4 9   2 4 9

5 4 7
5 - 0

4 5 6
4 2 e
```

a screenshot of the Multiple Segment Analysis Task in action.

As with the axis-dyad chord creation algorithms, we use a brute-force algorithm to accomplish this task, for many of the same reasons. That is, for each segment that is entered, we check every possible configuration, as we did in the statistical analysis described earlier. For each segment we then have a set of possible arrays. We then simply compute the intersection of all of these sets, leaving only those arrays that are compatible with all segments. This approach was used after considering a more complicated approach using graphs. However, before trying this approach we implemented the brute-force approach to see how it would perform. It turns out that this approach is quite adequate. We ran tests on several sets of up to 30 segments, ranging from randomly generated segments, which result in few arrays, to sets of segments that are all subsets of {1, 2, 3, 4, 5, 6}, which have many arrays in common. In all cases, the arrays are generated in under 10 sec. This seems fast enough, so we abandoned a

Figure 9. Arrays and chord segment configurations. (Mm.1–2 of Etude No. 1, from Six Etudes for Piano by George Perle. Copyright © 1978 [renewed] by G. Schirmer, Inc. [ASCAP]. International copyright secured. All rights reserved. Used by permission.)



more complex analysis of the problem that would lead to a more complex algorithm that may or may not be faster.

Analytical Applications

The program's speed greatly facilitates the preliminary analytical process of determining arrays and chord configurations. Entering the first six notes from Figure 9 in the Multiple Segment Analysis Task, but without specifying the interval cycles, generates 720 arrays. To decrease the unwieldy number of possibilities, the analyst specifies that interval 1 and 3 cycles are in operation, based on the observation that the first six notes (identified as segment A) form a palindromic succession of intervals <<1-3-3-3-1>>. As a result, the number of possible arrays trims down to just 16. The analyst then enters the notes from the next segment B, reducing the possible arrays to just two, namely 1,2/0,3 and 2,3/0,3. The analyst continues to enter segments one at a time to ascertain finally which of these arrays underlies the passage.

Detecting relationships among segments is also made easier with the Multiple Segment Analysis Task. The analyst can highlight the transpositional relationship of a descending major second between corresponding segments E-H, F-I, and G-J in measure 2 by choosing transpositionally related arrays and axis-dyad chord configurations.

T3RevEng is especially helpful in forming interpretations that are not so readily apparent. The excerpt in Figure 10a begins with segments from the array 8,9/6,9, determined by entering the segments one at a time. The nine dyads of measure 25, however, cannot be shown to emanate from a single array when considered as three axis-dyad chords. Interpreting Mr. Perle's dynamic marking of *pp subito* at the last dyad of measure 25 as indicating a different musical gesture, the analyst includes the last dyad with the segments in measure 26 instead of those in measure 25. The resulting axis-dyad chords in measure 26 all belong to a new array, 5,6/5,8.

The remaining dyads of measure 25 form four tetrachords, all of which may belong to 144 arrays. To manage this ungainly result, the analyst must find some means to constrict the data. Entering the four tetrachords in the Sum Square Task reveals that they all have the sum pairs 8/6 and 9/5 in common. The tetrachords are then arranged to feature these sums in the same physical position for all four squares, as illustrated in Figure 10b. The analyst then interprets these tetrachords as constituting a transitional passage, with the sum pair 8/6 providing the link from array 8,9/6,9 of the preceding phrase to 5,6/5,8 of the following phrase. The remaining sum pair functions as a secondary link, with sum 9 leading from the former array, and sum 5 leading to the ensuing array.

On occasion, the analyst will choose to bypass obvious conclusions suggested by T3RevEng's findings in favor of more musically convincing readings. In Figure 11, the tetrachordal segment G is reinterpreted as pivoting to a different array, 9,0/5,t. However, this tetrachord, as well as the next two at H and I, could still be analyzed as belonging to the former array, 8,e/6,e. Based solely on the data supplied by the Multiple Segment Analysis Task, the onset of the new array would be located at chord J, because its pitch classes cannot be arranged to fit the preceding array. However, the analyst chose to locate the pivot chord at G for interpretative preferences T3RevEng cannot consider. Although both

Figure 10. (a) Tetrachords of m. 25 provide transition between arrays. (Mm. 23–26 of Etude No. 1, from Six Etudes for Piano *by George Perle. Copyright © 1978 [renewed] by G. Schirmer, Inc. [ASCAP]. International copyright secured. All rights reserved. Used by permission.) (b) Tetrachords of m. 25 arranged as sum squares.*

Figure 11. Tetrachordal segments of arrays 8,e/6,e and 9,0/5,t. (Excerpt of "There Came a Wind Like a Bugle," from Thirteen Dickinson Songs, Vol. 1 *by George Perle [piano only, third system and first half of fourth system]. Copyright © 1981 [renewed] by G. Schirmer, Inc. [ASCAP]. International copyright secured. All rights reserved. Used by permission.)*

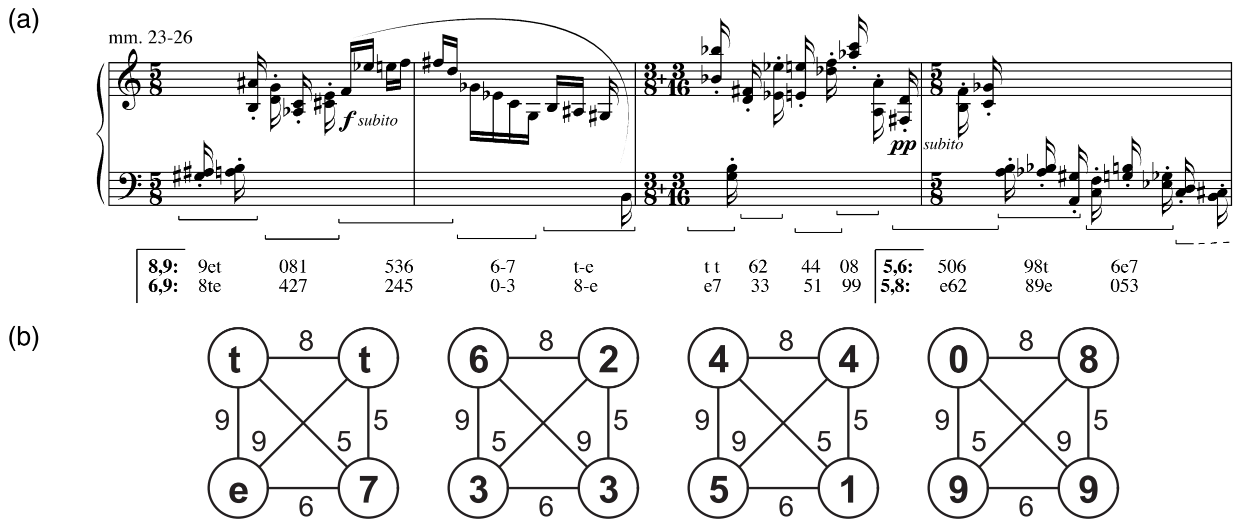


Figure 10

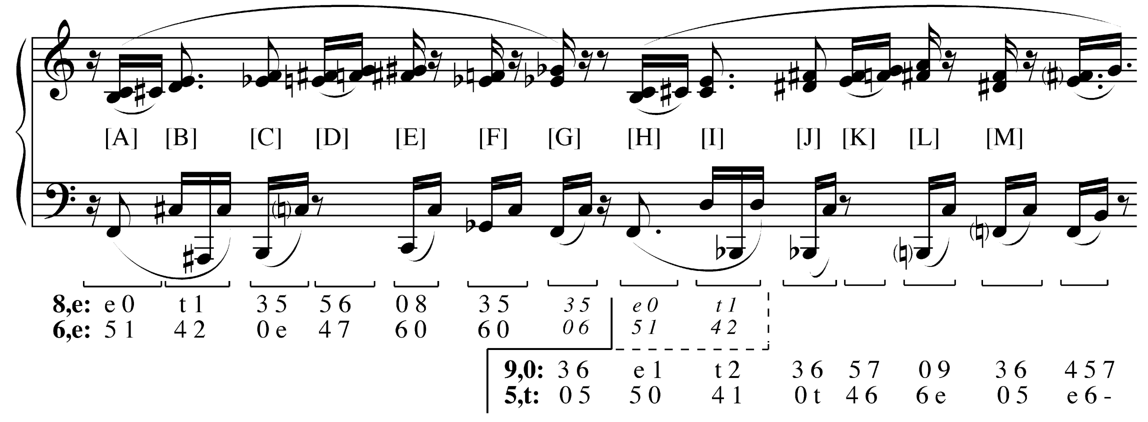

[A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M]
8,e: e 0 t 1 3 5 5 6 0 8 3 5 3 5 e 0 t 1
6,e: 5 1 4 2 0 e 4 7 6 0 6 0 0 6 5 1 4 2
9,0: 3 6 e 1 t 2 3 6 5 7 0 9 3 6 4 5 7
5,t: 0 5 5 0 4 1 0 t 4 6 6 e 0 5 e 6 -

Figure 11

tetrachords F and G share the same pitch-class content, the notes are rearranged at G, suggesting an appropriate location for the pivot. Furthermore, tetrachord H begins a new phrase. All the corresponding tetrachords in the two phrases are related symmetrically, with the first vertical dyad in each tetrachord remaining invariant, and the second vertical dyad undergoing a symmetrical transposition: The upper note of the dyad is transposed upward by a semitone (T+1), and the lower note is transposed downward by a semitone, (T–1). This is evident when comparing tetrachordal pairs A-H, B-I, C-J, D-K, E-L, and F-M.

Another situation requiring analytical intervention occurs in the opening measures of Mr. Perle's Invention from the *Wind Quintet No. 4,* given in Figure 12. This passage divides into two arrays, 9,e/6,1 and t,e/6,1, with a concurrent change of interval cycles. Three of the segments in this passage can be analyzed as belonging to both arrays, raising the question of where to locate the modulation from one array to the other in a musically satisfying

Figure 12. Possible pivot chord locations. (Mm. 1–4 of Invention, from Wind Quintet No. 4 *by George Perle. Copyright © 1987 by Galaxy Music Corporation. All rights reserved. International copyright secured. Used by permission.)*

Figure 13. Segments from array 9,3/6,1. (Mm. 1–4 of Invention, from Wind Quintet No. 4 *by George Perle. Copyright © 1987 by Galaxy Music Corporation. All rights reserved. International copyright secured. Used by permission.)*

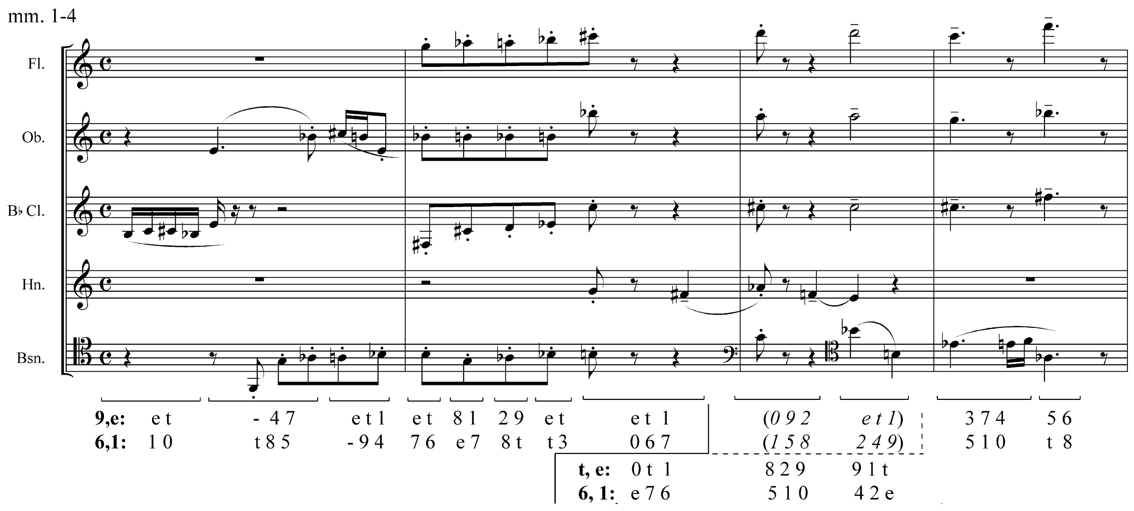

mm. 1-4
Fl.
Ob.
B♭ Cl.
Hn.
Bsn.
9,e: e t - 4 7 e t 1 e t 8 1 2 9 e t e t 1 (0 9 2 e t 1) 3 7 4 5 6
6,1: 1 0 t 8 5 - 9 4 7 6 e 7 8 t t 3 0 6 7 (1 5 8 2 4 9) 5 1 0 t 8
t, e: 0 t 1 8 2 9 9 1 t
6, 1: e 7 6 5 1 0 4 2 e

Figure 12

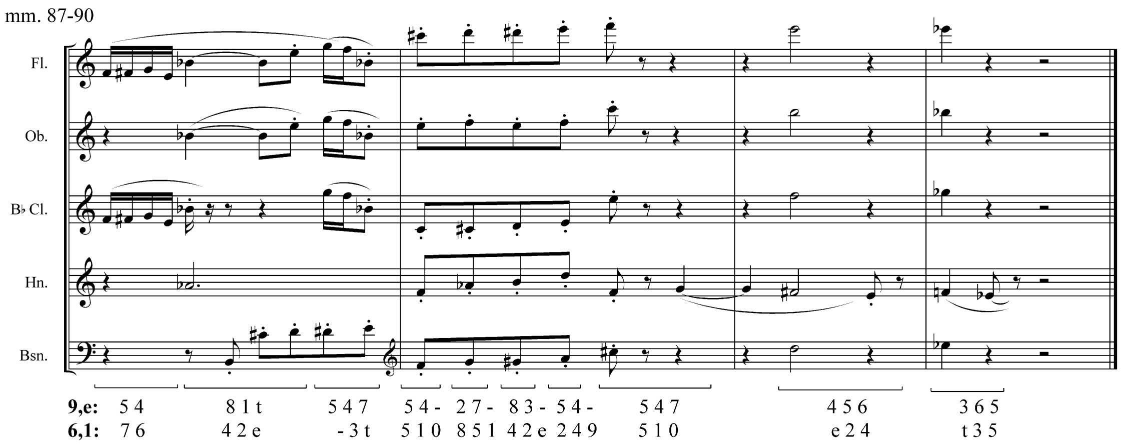

mm. 87-90
Fl.
Ob.
B♭ Cl.
Hn.
Bsn.
9,e: 5 4 8 1 t 5 4 7 5 4 - 2 7 - 8 3 - 5 4 - 5 4 7 4 5 6 3 6 5
6,1: 7 6 4 2 e - 3 t 5 1 0 8 5 1 4 2 e 2 4 9 5 1 0 e 2 4 t 3 5

Figure 13

way. In this case, the analyst interprets the modulation as occurring on beat three of the second measure, the first of the three potential pivot locations. Several factors contribute to this decision.

First, the initial statement of the motive/countermotive pair concludes on this beat. Second, the French horn makes its first entrance precisely at this point. Additional justification is provided by the relationship between the opening and closing measures of the movement. As with the passage from measure 1 shown in Figure 12, the passage from measure 87 begins in array 9,e/6,1, shown in Figure 13. The notes within the segments of measure 1 are transposed by six semitones (T6) in the corresponding segments in measure 87. The tetrachordal segments in measure 2 have expanded to pentachordal segments in measure 88 owing to the addition of the French horn, yet the pitch classes of the corresponding sums are also transposed by T6.

This tritone relationship holds until beat 3 in measure 88. At this point, the lower cyclic set does not maintain the T6 relationship; instead, it is symmetrically related to its counterpart, with the neighbor notes transposed up by five semitones (T+5) and the axis note transposed down by the same amount (T–5). Also from this point, the five instrumental parts are rhythmically modified in comparison to their setting in measures 3–4.

Future Work

An obvious question that comes to mind is to what extent a computer can be used to help analyze music written using Mr. Perle's compositional system of twelve-tone tonality. We have demonstrated in the present work a program that can compute useful information given a sequence of chord segments, performing in seconds what could take weeks or months for a human analyst. The natural question to ask is how much more can be automated, and what tasks absolutely require the expertise of a human analyst.

Currently, the analyst must break up the composition into segments by hand. We would like to investigate whether an efficient algorithm exists to segment a composition, and even make some tentative decisions about where the array names might change based on less input from the analyst. At present, the analyst uses cues from the music—including changes in meter, instrumentation, contour, texture, dynamics, articulation, rhythmic patterns, tempo, pedal markings, nonmetrical beaming assignments, and a myriad of other indicators—to create segments. An algorithm can take some of these things into account if the composition is input in a suitable format. For example, the notes can be input so that their location in the score and their voice or instrumental designation is indicated, allowing the algorithm to take this into account when deciding segments or array name changes.

There are other things, like contour, syncopation, or nonmetrical beamings, that an algorithm cannot grasp so easily. Although a computer program is limited in that it has no intuition or knowledge of music theory, it can consider many possibilities much faster than a human analyst. What is unclear at this point is whether we can find a suitable algorithm that is fast enough to overcome the lack of intuition. When working with segments of fixed sizes (three to six in the current work), the number of possible arrangements is small enough to deal with. However, if an algorithm is given an entire composition, there are so many possible ways to segment it that it would not be possible to do an exhaustive analysis, even with a computer. We must find some heuristics to narrow down the segmentations that are considered and to determine possible array names and transitions.

We also need to determine exactly what inputs are needed. Should we input the exact notes or just pitch classes; do we need the durational value of the notes, or just the pitch information; how should various articulation markings be dealt with; how do we specify parts, and do we need to; and do we provide as input a list of possible interval cycles or array names to be considered? The more we can supply as input, the better an algorithm can perform, but also the more manual work the analyst must perform.

Although it is certain that a program will not replace the human analyst, we may be able to create a program that can produce several (not too few or too many) segmentation alternatives for the analyst to consider. This will allow the analyst to spend less time on rote computations and more time on the interpretative questions generated during and by the analytical process.

Final Observations

The T3RevEng program is a calculator designed solely to generate data, and as such requires an individual trained in twelve-tone tonality to interpret the information it yields. When understood and used in this sense, T3RevEng has proven to be a virtually indispensable tool for the analyst of Mr. Perle's twelve-tone tonal music. (It must be stressed that this application does not analyze phrases in a musical way and does not solve interpretative quandaries.) T3RevEng may also be useful for composition within the system of twelve-tone tonality. The composer may enter input from a variety of perspectives, including potential pitch class segments, or desired interval cycles or tonic sums, and the application will provide as output a wealth of information for the composer's consideration. In short, the program can supply the composer with a comprehensive set of pre-compositional material, based on the composer's specifications. As such, then, the T3RevEng program is a practical, robust tool for both analysts and composers working within the parameters of twelve-tone tonality.

References

Carr, J. 1995. "George Perle and the Computer: An Uneasy Alliance." *International Journal of Musicology* 4:207–215.

Carrabré, T. 1993. "Twelve-Tone Tonality and the Music of George Perle." Ph.D. dissertation, City University of New York.

Foley, G. 1999. "Pitch and Interval Structures in George Perle's Theory of Twelve-Tone Tonality." Ph.D. dissertation, University of Western Ontario.

Forte, A. 1973. *The Structure of Atonal Music.* New Haven, Connecticut: Yale University Press.

Headlam, D. 1995. "Tonality and Twelve-Tone Tonality: The Recent Music of George Perle." *International Journal of Musicology* 4:301–333.

Lewin, D. 1987. *Generalized Musical Intervals and Transformations.* New Haven, Connecticut: Yale University Press.

Perle, G. 1977. *Twelve-Tone Tonality.* Berkeley: University of California Press.

Perle, G. 1996. *Twelve-Tone Tonality,* 2nd ed. Berkeley: University of California Press.

Rahn, J. 1980. *Basic Atonal Theory.* New York: Schirmer.

Rosenhaus, S. 1995. "Harmonic Motion in George Perle's *Wind Quintet No. 4.*" Ph.D. dissertation, New York University.

Straus, J. 2005. *Introduction to Post-Tonal Theory,* 3rd ed. Upper Saddle River, New Jersey: Prentice-Hall.