victor burton

All-Combinatorial Hexachord Tester

MIRROR INVERSION, which will be our chief concern in this study, is a phenomenon not confined music. It is a structural form *in nature* itself; therefore its appearance in music should not surprise no one.[1]

We, human, thinker, creator, composer have always been drawn towards new ways to organize or conceptualize the various components of a given system. Western music tradition makes no exception. At the turn of the 20th century, interest for a refreshing way of organizing harmony and melody arose: twelve-tone technique, or dodecaphony. Leaving out and even avoiding any reference to the codes of functional diatonic music, this radical approach steered abruptly our view of music. The very first individuals that delved into this realm of new possibilities are Charles Ives, Josef-Mattias Hauer and of course the famous Arnold Schoenberg. Although those composers left wonderful and intricate music behind, they did not explicitly discussed nor documented some specific theorical concepts. One of them is around the issue of constructing a distinct type of twelve-tone row that is characterized by its *all-combinatoriality*.

All-Combinatoriality: *the capacity of a collection to create aggregates with forms of itself and its complement under both transposition and inversion.* (Babbitt, 1987).

This very topic is the one I tackled in my project. More precisely, my objective was to create a C++ program that could test an hexachord entered by the user and output if true/false the given hexachord adheres to the all-combinational properties covered and if so with which point(s) of inversion.

The core reference for this project was “The Hexachord and its Relation to the 12-Tone Row” by George Rochberg (1955, Theodore Presser Company). In this book, Rochberg dissects the subject of all-combinationality when building twelve-tone rows. He proceeds empirically by firstly demonstrating assorted rows taken from various works by Schoenberg, then gradually identifying patterns or laws that he finally consolidate in a set of 3 clear rules. Those rules are the following:

RULE 1. Any one horizontal interval plus the mirror interval (or point of inversion) cannot equal 12 (or a multiple of 12).

RULE 2. Any two horizontal intervals plus mirror interval cannot equal 12 (or a multiple of 12).

RULE 3. Any double one interval plus the mirror interval must not equal 12 (or a multiple of 12).

1 Taken from the Foreword of Rochberg’s book.

Notes about hexachordtester.cpp:

To run the program, the user must first compile the file then run the program entitled "hexachordtester" with the desired hexachord included in the same command, as follow:

```
./hexachordtester 0,1,2,3,4,5
```

If the user inputs more or less than 6 pitch class, the program will notify him/her.

The C++ file should be self-explanatory. Fundamentally, the formulas representing the three rules are fairly basic. I left comments either pointing a meaningful component/element or explaining how the structure was envisioned. What follows is extra information regarding certain architectural decisions and then suggest improvements for a future update.

About order of the rules: in the code, the rules are processed (tested) in the order of rule 1, rule 3 and rule 2. This was a suggestion from Sevag, who is a professional system-reliability engineer. I am mentioning his title because only someone who has considerable experience in large-scale projects will have the thought of ordering the different operations by their computational cost from cheap to high. In fact, we can easily conceive that:

RULE 1 is the cheapest of all, only executing an addition between the points of inversion and the 6 notes of the hexachord

RULE 3 is following by executing a simple multiplication (x2) of every 6 notes of the hexachord followed by an addition with the points of inversion.

RULE 2 is the most expensive, by having to test both the vector of the hexachord with itself but also with the "mirrored hexachord" (again with itself), added to the points of inversion.

As it is in the current file, the very moment the hexachord does not respect a rule, a "false" message is instantaneously registered in relation to the given point of inversion and the following rule(s) are bypassed, thus sparing the computer from superfluous operations. However, in the case of small fixed size vectors and powerful modern computer, this optimization is not tangible. Also, this method has the slight downside that the user cannot know and retrace *which* rule was not adhered to. For better clarity, I could adjust the program to output the result of each rule for every point of inversion instead of a generic "fail/succeed" message.

About RULE 3: towards the end of his book, Rochberg converts RULE 3 in a sort of premise. In fact, one can quickly comprehend that any even-numbered point of inversion added to any duplicated note will soon or later = 12 (or a multiple 12). Inversely, one can perceive that any odd-numbered point of inversion added to any duplicate note will *never* = 12 (or any multiple of 12), as $x*2 + y_{odd} \neq 12$ (or any z_{even}). This explains why I hard-codded the points of inversion 1, 3, 5, 7, 9 and 11 in the vector holding the points of inversion. Thus, RULE 3 is presently never executed and is redundant. If desired, one could remove the fixed value and let the program test the program for every points of inversion between 1 and 11, including even numbers, which will make veritable use of RULE 3.