Computer Aided Analysis of Post-Tonal Music
by Michael Schutz

1) Foreword

The following article represents my work in developing software to assist in the analysis of music. Specifically, my software is designed to analyze post-tonal music; however it could just as easily be adapted to perform analysis of a different kind. My aim was not to create a specific program, but rather the general toolset needed for future work in using computers as analytical aids.

While music has always displayed striking mathematical patterns, this relationship has been taken to new levels of complexity in the 20th Century, and the possibilities of automating portions of the analytical process are as limitless as the analytical techniques themselves. By taking advantage of the ever-increasing power (and decreasing cost) of today's computers, musicians have a new opportunity to apply the speed and persistence of personal computers to some of music theory's most tedious and repetitive calculations. My primary goal with this project was to lay the foundations for the development of future software capable of assisting in analytical tasks currently performed completely by hand.

Automating a process as intricate and complex as analysis of music was an elaborate undertaking. Most of my time was devoted to developing the needed foundations for such a project. Building upon

Michael Schutz received bachelor's degrees with honors in music performance and in computer science from Penn State University in 2002. He is currently pursuing a master's degree in music technology at Northwestern University. For further information about Mr. Schutz's ongoing project, email him at mschutz@northwestern.edu.
this, I then set out to demonstrate the possibilities of automated assistance in performing analyses by implementing a simple searching routine based upon interval-class vectors. Given the foundations needed for this project, it will now be possible to develop more sophisticated tools for assisting the analytical process.

This software represents a sizeable programming project, with the bulk of my time dedicated to designing and implementing over 22,000 lines of code. Yet the project is by its very nature a musical one, and in that sense the technical details of its implementation are less important than its musical capabilities. Therefore this article is primarily focused on the musical possibilities introduced by my software. For compactness and clarity, I have endeavored to leave technical details to a minimum (with the obvious exception of chapter 3, on the VirtualScore). For those unfamiliar with MIDI terminology and/or 20th Century analytical techniques, I have included a glossary of important terms in Appendix 2.

Note: There is debate among musicians of all levels regarding the benefits of using pitch-class analysis to describe music, and I am not attempting to take sides in this discussion. I have demonstrated the possibility of using computers to assist with musical analysis by implementing one well-established analytical technique. While there can be lively debate as to the musical significance of my software’s findings, this illustrates the very heart of what my software accomplishes - it offers musicians a tool for examining, dissecting, discussing, and better understanding music.

It would not have been possible to reach this point without significant help from several individuals and institutions. Grants jointly awarded from the Penn State Schreyer Honors College, School of Music, Department of Computer Science and Engineering, and College of Arts and Architecture provided much needed funding to hire several students as testers at various stages of the software’s development.

John Hannan (Penn State Department of Computer Science and Engineering) and Paul Barsom (Penn State School of Music) jointly served as advisors, and offered a great deal of assistance in their respective areas. Both were helpful and supportive, and both had enormous influence on the outcome of my work.

2) Introduction

The twentieth century has seen astounding leaps in the compositional complexity of its music. Contemporary theory has evolved with advances in compositional intricacy, and serves as an integral part of appreciating and comprehending contemporary repertoire. However, increases in compositional complexity have resulted in an intricate, time-consuming, and at times tedious analytical process. The plethora of arithmetic calculations involved in counting notes, normalizing pitch-class sets, and calculating interval-class vectors serve to hinder the speed with which one can draw meaningful conclusions from analysis. This strongly suggests a push towards automation, which holds the potential to improve the speed, accuracy, and scope of any in-depth musical analysis. While a computer’s potential for rapid examination is limited by its lack of intuition, it offers the perfect complement for any well-trained musician.

The inspiration for this research came from personal frustrations with understanding post-tonal works within my own repertoire. While preparing for an upcoming performance of contemporary music, it became apparent that although there was a definite ordering to the piece’s pitch content, this ordering was not easily definable. After pouring over the piece in greater detail, I realized that even though analysis was essential in comprehending the overall pitch structure, inspecting a score note-by-note was a tiresome and time-consuming task. The more I continued, the more it seemed this type of analysis was, at the lowest level, a series of monotonous and repetitive calculations. In other words, the perfect task for a computer.
The software I have developed is capable of reading music encoded in current industry-standard means of digitally representing music, translating it into a VirtualScore (a new file format I developed specifically for analysis as a part of this project), and then searching it for musical patterns. After picking out groups of notes to examine, my software is capable of using a pre-determined evaluation metric to gauge the musical similarity of each note grouping. By exhaustively scanning all possible note combinations in a musical score, my software is able to rapidly and accurately locate user-defined musical patterns, and report on their frequency and location. This is in striking contrast to the length of time a comparable analysis would take by hand, and offers a higher degree of accuracy. By testing all possible combinations of notes, my software is able to guarantee that patterns do not go undetected, and offers a powerful search tool to any musician desiring a better musical understanding of pitch structure.

Automation frees the theorist from repetitive tasks, enabling him/her to reach meaningful conclusions more quickly. In seconds, my analysis software can locate potential note groupings that would take hours to select by hand, opening up new doors for large-scale investigations of pitch organization. Grouping multiple compositions sharing a common composer, style, or time period promises the possibility of examining similarities (or differences) in pitch structure in a manner simply impractical if done by hand. By handling the bulk of the repetitive arithmetic calculations associated with such an analysis, my software allows musicians to focus on musical meaning, rather than mathematical manipulations.

2.1) Overall Goals

The goal of this research was to explore the possibilities afforded by automating the analytical process, serving as a tool for musicians interested in increasing their musical understanding. It will aid (in both speed and inspectional power) efforts to comprehend contemporary music. Like any musical tool, it does not negate the importance of a strong musical background, but rather serves as a guide, pointing out relevant musical facts. Much as a metronome is a helpful tool in developing, but a poor substitute for, a strong sense of time, this tool is helpful in developing an understanding of pitch structure in post-tonal music. This technology can aid musicians in revealing new insights about the music we analyze and perform, which will serve to increase understanding and appreciation among theorists and performers alike.

Rather than create software useful for a single project, or even a single analytical operation, my goal was more broad— to create the necessary tools to facilitate automated assistance of the analytical process itself. The software designed for this project will function well as part of a larger system for musical analysis. The VirtualScore was designed specifically to serve as a viable format for automating virtually any analytical operation. All possible information is encoded into a VirtualScore, including those parameters that are not of immediate interest when examining only pitch structure. Although currently only pitch information is directly used for analysis, information regarding rhythm, temporal location, orchestration, and dynamics are included in a VirtualScore for future use in automating other analytical endeavors.

2.2) Overview

Reading a Standard MIDI File, my software is able to translate musical information into another specialized representation simulating an actual musical score. This VirtualScore can then be examined much the same as its real-world counterpart, allowing for automated analysis analogous to that performed by any musician. This offers the possibility of identifying millions of potential pitch-class sets per second, yielding exhaustive examinations previously impractical. Designed for maximum usability, the program is implemented entirely in Java ensuring compatibility with any platform on any computer, including the potential for future use on handheld Personal Digital Assistants (PDAs).
Once music is encoded in a VirtualScore, it is possible to view previous/ successive notes (i.e. examine a line melodically), or notes sounding simultaneously (harmonic analysis). This ability is crucial as it parallels methods of analysis currently performed by hand. Once a group of notes is defined as having some relationship, this relation is formalized and can be evaluated using any of several metrics. For instance, taking the prime form or interval-class vector of a pitch-class set gives a common basis to manipulations and allows for identification of recurring pitch material.

Following practices of traditional manual analysis, individual notes defining a pitch-class set can be required to occur within certain proximity. This ensures grouped notes contain an aurally perceptible relationship, and precludes an excessively mathematical analysis overstepping the bounds of audible reality. Comparison and interpretation of the results of various operations will yield a plethora of data from which one can draw musically meaningful conclusions.

3) The VirtualScore

Design of the VirtualScore represents a significant portion of the work involved in enabling automated assistance of the analytical process. The VirtualScore was designed specifically to enable automated analysis, and offers several features crucial to this task. Compatibility with MIDI allows for music input from Finale, Sibelius, or any program supporting Standard MIDI. The nature of the VirtualScore also allows for compatibility with other musical formats that may gain popularity in the future. A provisional patent for the VirtualScore was filed with the United States Patent and Trademark Office in April of 2002.

This chapter deals exclusively with the motivation, design, and use of the VirtualScore, and is intended for those interested in its technical details. Those interested only in the musical applications of the VirtualScore are encouraged to skip ahead to chapter 5, "Musical Example," for information regarding the musical applications of the VirtualScore format.

3.1) Introduction

For maximum flexibility, and to provide the opportunity to create VirtualScores from any specifications, the VirtualScore is designed to be completely independent of the file format from which it is translated. Rather than building a VirtualScore directly from MIDI information, data is first organized into a collection of Note objects. Each musical note is represented as a Note object, which contains fields describing a note's start time, length, attack velocity, release velocity, and the score line on which the note occurs. Later this raw mathematical data can be translated into musically meaningful measure and beat numbers describing a note's start and end times, rhythm, dynamic, and sounding instrument. Each note is then individually inserted into the VirtualScore. A VirtualScore can be built from any file format, provided the format includes a minimum of needed information. The design of the VirtualScore provides an abstraction from the source file's implementation, as well as a mechanism for overcoming the source file's flaws.

The remainder of this chapter details the motivation, design, implementation, and applications of the VirtualScore. Section 3.2 outlines generic requirements for performing automated analysis of music, derived from the mechanisms by which analysis is currently performed manually. Sections 3.3 and 3.4 review the Standard MIDI File format's features, and why the VirtualScore is well suited to analytical endeavors. Sections 3.5-3.8 discuss the features, format, and process of creating a VirtualScore, including a short demonstration of the principles governing its design (Section 3.6).
3.2) Definition of requirements

The six main criteria a format must offer in order to enable effective analysis are:

1) Fast, random access to any note (regardless of location)
2) Ability to search notes vertically (i.e. examine harmony)
3) Ability to search notes horizontally (i.e. examine a melody)
4) Temporal relationships between each note and entire score
5) Pitch, rhythm, location, dynamic, and instrument information for each note
6) Ability to recognize/ notate complex musical passages

These six requirements serve as benchmark requirements for performing automated analysis. It is essential that any program attempting to automate the analytical process support all of them efficiently.

3.3) The Standard MIDI Format (SMF), Pros and Cons

While SMFs are great for facilitating inter-software/ inter-instrument communication, SMFs are neither designed nor appropriate for facilitating computer-aided analysis. Due to the physical organization of an SMF, it may be necessary to traverse large amounts of data in order to locate a single note. (See Appendix 1 “Illustration of SMF’s Shortcomings,” for a detailed example). This inability to retrieve individual notes efficiently makes it difficult to select and view notes within a SMF. Notes at the end of a large file will take longer to retrieve than notes near the beginning, severely slowing the analytical process. Difficulties involved in selecting notes translate directly into problems analyzing harmony and melody (requirement 3).

The lack of a time stamp associated with MIDI messages makes analyzing temporal relationships between notes quite difficult. While notes melodically adjacent are often placed physically adjacent, this cannot be assumed across all files. Notes aligning harmonically among different instruments will rarely share any reliable physical proximity. These two factors translate into search algorithms that must scan entire files to locate notes sounding at any particular point. Given that most analysis requires repeated selecting of particular notes from a score, this is an extremely ineffective method of searching even moderately sized files. Lastly, SMFs are restricted to representing only 16 unique instruments at a given time, which serves as a severe hindrance when representing large-scale compositions.

3.4) The VirtualScore Solution

Despite its many limitations, it was imperative that the VirtualScore be compatible with the SMF format, as practically every notation program, software sequencer, and musical hardware currently available supports it. The solution to conflicting requirements for accessibility and capacity for analysis was twofold. First, I created the VirtualScore, a new file format designed specifically for musical analysis. Second, I implemented a translation algorithm to convert information from an SMF format into an equivalent VirtualScore. This allows for the best of both worlds - fulfillment of all requirements for analysis along with accessibility from Finale, Sibelous, or virtually any notation program. As the VirtualScore was designed to be completely independent of the format originally used to encode the musical information, in the future it can easily be created from any other type of digital musical representation.

The VirtualScore is able to overcome limitations in ensemble size and complexity, and promises compatibility with any future enhancements to the analysis components of the system. Along with random access, the VirtualScore also offers the ability to examine music melodically (e.g. successive notes performed on the same
instrument), and harmonically (e.g. notes played within close time proximity on any number of instruments). Using the information taken from the SMF, the VirtualScore can even distinguish instruments (or hands of a piano part), and allow for examinations confined to particular instrument groupings.

The VirtualScore satisfies previously conflicting demands for simultaneous flexibility and accessibility. Although currently the software only examines pitch aspects of compositions, the flexibility of the VirtualScore means that practically any type of analysis could be performed in the future, depending upon the interests of users.

3.5) Advantages of the VirtualScore

The VirtualScore overcomes all of the above limitations of SMFs. In a VirtualScore, each note is represented as a time-stamped Note object, which allows for its context to be readily understood. Complex music demands powerful representational ability. The VirtualScore offers no restriction on ensemble size, which is helpful when analyzing compositions using many simultaneous instruments. All information regarding pitch, rhythm, start/ stop time, dynamic, and instrument encoded in an SMF is available in the VirtualScore, however in a manner much more conducive to analysis.

In addition to meeting the abovementioned requirements for analysis, the VirtualScore also includes other capabilities important in displaying the results of analysis. While it is possible to derive much of the information contained in a VirtualScore from an SMF, some is not explicitly defined (e.g. timestamps, instrument association, rhythmic recognition), and requires a great deal of calculating to derive. In addition to the information conveyed in a VirtualScore regarding raw mathematical values describing pitches, dynamics, note length, and time stamping information, the VirtualScore is capable of interpreting this information and displaying it in a format useful for musicians.

Whereas SMFs do not sufficiently satisfy any requirements for facilitating computer-aided analysis, the VirtualScore fulfills all of them. It is worth noting this shortcoming on the part of SMFs is not a flaw in their design. The SMF format was never intended for analysis, and it is understandable that when applied to such a situation it proves inadequate. It is a testament to its strength that it remains a viable, industry-standard means of digitally representing music over twenty years after it was first defined.

3.6) VirtualScore Demonstration

The VirtualScore emulates piano-roll representations of music popular in most sequencers. This offers the ability to quickly and efficiently determine all notes sounding at a given point in time. To demonstrate this ability, a simple melody is shown below, first in traditional notation, and second with piano-roll style notation.

Simple melody as represented in traditional notation

The same simple melody, represented in piano-roll style of notation.
By examining these two representations of the same melody, it is possible to see the relationships between the two formats. In piano-roll notation, each note’s horizontal position defines its starting time, the length of its sounding time, and the vertical position of its pitch. For example, the first note indicates the pitch C3 sounds for one quarter note, beginning on beat 1 and ending on beat 2. Likewise, the piano-roll clearly shows two pitches sounding on beat 3 (pitches C3 and G3).

Note the ease with which it is possible to determine notes sounding at any given point (which includes both those notes started on the given beat, or those still sounding from a previous articulation). The above melody demonstrates this property when examining the notes sounding on beat 4. Although it is obvious to musicians reading traditional notation that the notes C and G are still ringing (despite the lack of an explicit notation on beat 4), enabling a computer to make such a seemingly trivial observation is a surprisingly involved process. This points to a larger problem with traditional notational systems as well as the SMF format: in order to find all notes sounding on a particular beat, it is necessary to examine every previous note in order to see whether it is still sounding. Musically, it is quite possible a note that began sounding several beats ago may still be ringing. Mathematically, it is possible a note that began in the first measure may still be sounding. With traditional notational practices, this is addressed by redrawing a note at the beginning of each written measure (with a tie to the previous articulation). With a piano-roll representation of music it is clear from looking vertically down a score exactly which notes are sounding at any given time.

For example, in the above representation, if the user wishes to examine all notes sounding on beat 4, a query to the VirtualScore to that effect would result in a simple scan down the score on beat 4, in a fashion somewhat like:

Such a query would clearly indicate the notes C3 and G3 (and only those notes) are still sounding, even though they began ringing on beat 3.

In contrast, such a query to the same melody in a format more resembling traditional notation would have a more problematic result, as nothing is explicitly notated on beat 4.

Are notes sounding on beat 4?

3.7) Viewing the VirtualScore

*Measures 1-5 of Arnold Schönberg No 1 from* "3 piano pieces, op. 11" */UE 2991*

©1910 by Universal Edition A.G., Wien, renewed ©1938 by Arnold Schönberg
Finale® representation of actual score

Printout of VirtualScore of Op. 11, No. 1:

Long ringing notes intentionally occur multiple times in the VirtualScore printout. Printed below are all times at which a note begins sounding. Notes are displayed whenever note attacks begin. Longer ringing notes are also displayed (with an asterisk) at the attack points for other notes. For example, G3, which is still sounding on beat 2 of measure 2.

Also, as only analysis of post-tonal music is currently implemented, the VirtualScore assumes enharmonic equivalence. Therefore the note printed as a G# in measure 1, beat 3 of the sheet music is identified as the enharmonically equivalent Ab in the following printout:

Measure 1:
1:2) B3 (Quarter Note, <Midi Ch: 1>)
1:3) Ab3 (Quarter Note, <Midi Ch: 1>)

Measure 2:
2:1) G3 (Dotted Quarter Note, <Midi Ch: 1>)
2:2)
   *G3 (Dotted Quarter Note, <Midi Ch: 1>)
   B2 (Half Note, <Midi Ch: 1>)
   F2 (Half Note, <Midi Ch: 2>)
   Gb1 (Half Note, <Midi Ch: 2>)
2:2.5)
   A3 (Eighth Note, <Midi Ch: 1>)
   *B2 (Half Note, <Midi Ch: 1>)
   *F2 (Half Note, <Midi Ch: 2>)
   *Gb1 (Half Note, <Midi Ch: 2>)
2:3)
   *F3 (Quarter Note, <Midi Ch: 1>)
   *B2 (Half Note, <Midi Ch: 1>)
   *F2 (Half Note, <Midi Ch: 2>)
   *Gb1 (Half Note, <Midi Ch: 2>)

Measure 3:
3:1) F3 (Half Note, <Midi Ch: 1>)
3:2)
   *F3 (Half Note, <Midi Ch: 1>)
   Db3 (Half Note, <Midi Ch: 1>)
   A2 (Half Note, <Midi Ch: 2>)
   Eb1 (Half Note, <Midi Ch: 2>)
3:3)
   E3 (Quarter Note, <Midi Ch: 1>)
   *Db3 (Half Note, <Midi Ch: 1>)
   *A2 (Half Note, <Midi Ch: 2>)
   *Eb1 (Half Note, <Midi Ch: 2>)
3.8) SMF to VirtualScore Conversion

The creation of the VirtualScore involves several stages:

1) Scanning through the input file, grouping together related bytes to recognize MIDI messages
2) Matching MIDI note-on messages with corresponding MIDI note-off messages to form Note objects
3) Stamping a start and end time into each Note object (thereby defining its length, which can later be interpreted as a rhythm)
4) Inserting each Note object into a VirtualScore

This score building process is best represented below:

As each MIDI note-on message is encountered, it is passed to the NoteHolder, an object designed to pair up note-on and note-off messages. Each note-on message is saved until the corresponding note-off message is encountered. Given a note-on/note-off pairing, the NoteHolder creates a new Note object (complete with note index, start time, end time, attack velocity, and release velocity information). In this way, MIDI messages are matched together to create Note objects. Each Note object contains the information found in both its note-on and note-off MIDI messages, making an examination of the Note at a later time much easier than in an SMF (where finding a note's length would require traversing an undetermined portion of the entire file). In addition to basic MIDI information, each Note object contains a timestamp indicating when it occurs within the VirtualScore.

In an SMF, timing messages are interspersed with MIDI messages as a means of indicating elapsed time within a MIDI file. In keeping with the MIDI standard, no inherent time is associated with each MIDI message. To associate a given message with an absolute time of occurrence, each timing message must be identified and added to a total running time counter. Each Note object is stamped with the time at which it occurs, thereby associating previously temporally indifferent MIDI messages with a specific timestamp. Each Note object is then inserted into a VirtualScore.

Once created, a VirtualScore can be analyzed in a manner similar to current methods of manual analysis. The VirtualScore allows one to view notes melodically as well as harmonically, and offers random access to any note. As every note is time stamped, it is possible to locate groups of notes sounding sequentially, simultaneously, or in any desired sequence. The wealth of information available for each note (pitch, rhythm, location, dynamic, and instrument) enables a variety of analytical operations. The VirtualScore also overcomes the SMF format's sixteen-instrument limitation, offering the ability to represent complex music for any instrumentation, with any combination of time signatures, tempo changes, and/or key shifts.
4) Process of Analysis

After creating a VirtualScore, the possibilities for automated assistance of analysis are limitless. Included in analysis software components are mechanisms for selecting note groupings, examining note groupings, and recording the results of such an examination. This includes customizable user-defined search constraints affording control over the proximity within which notes must occur in order to be grouped, the extent of a search, and the specific tools used to evaluate selected note groupings.

4.1) Description

Analysis is performed by scanning through a VirtualScore, selecting initial note groupings, and then successively examining all possible combinations of the selected notes. As an example, a simulated analysis is performed below. The following simple melody will reveal, step-by-step, the process by which a musical example is examined for a given pattern. For purposes of this demonstration, the following assumptions will be made.

1) The melody below is already represented by a VirtualScore
2) The interval-class vector to be located is \((0,1,0,0,0,0)\)
3) The proximity with which notes must occur in order to be grouped is defined as 2 beats

4.2) Simulated Analysis

```
musical example on which analysis is performed
```

First pass:

<table>
<thead>
<tr>
<th>Step</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Select the first note grouping (notes between beats 1 and 3)</td>
<td>C3, C3, C3</td>
</tr>
<tr>
<td>2) Build all possible 2 note combinations from this group</td>
<td>C3, C3, C3, C3, C3</td>
</tr>
<tr>
<td>3) Convert each note grouping into a set of pitch classes</td>
<td>C, D, C, E</td>
</tr>
<tr>
<td>4) Calculate interval-class vectors of these combinations</td>
<td>C, D, E, D, E, C, E, D</td>
</tr>
<tr>
<td>5) Check interval-class vector values against the desired interval-class vector</td>
<td>C, D, E, D, E, C, E, D</td>
</tr>
<tr>
<td>6) If there is a match, check to see if this note grouping is already recorded</td>
<td>C, D, already found? (yes)</td>
</tr>
<tr>
<td>7) If note grouping has not been recorded, then record it; otherwise, ignore</td>
<td>record C, D</td>
</tr>
</tbody>
</table>

Second pass:

<table>
<thead>
<tr>
<th>Step</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Select the second note grouping (notes between beats 2 and 3)</td>
<td>D3, E3, C3, C3</td>
</tr>
<tr>
<td>2) Build all possible 2 note combinations from this group</td>
<td>D3, E3, D3, E3, E3, C3</td>
</tr>
<tr>
<td>3) Convert each note grouping into a set of pitch classes</td>
<td>D, E, C, D, G</td>
</tr>
<tr>
<td>4) Calculate interval-class vectors of these combinations</td>
<td>D, C, D, G, D, G, C, G, E</td>
</tr>
<tr>
<td>5) Check interval-class vector values against the desired interval-class vector</td>
<td>D, C, D, G, D, G, C, G, E</td>
</tr>
<tr>
<td>6) If there is a match, check to see if this note grouping is already recorded</td>
<td>D, C, already found? (yes)</td>
</tr>
<tr>
<td>7) If note grouping has not been recorded, then record it; otherwise, ignore</td>
<td>record D, C</td>
</tr>
</tbody>
</table>

Computer Aided Analysis of Post-Tonal Music
This process is continued throughout the remainder of the composition. By trying every possible mathematical combination of notes in turn, my software is able to guarantee any pattern meeting the search criteria is identified and recorded regardless of where it occurs within a composition. By comparing multiple scores, it is possible to search for recurring patterns throughout several compositions.

After using this software to search exhaustively for all occurrences of a given pattern, it may be necessary for a musician to read through the results and discard certain findings. Even notes occurring within a fairly short amount of time (e.g. 2 beats), may share little perceptible relationship depending upon the number, volume, and range of notes in between. My software will never function as a replacement for an understanding of musical theory, or as a replacement for familiarity with the music analyzed. Rather, it is a powerful tool for use by those already well versed with the analytical processes seeking to analyze more efficiently.

5) Musical Example

To illustrate the results of this software, I have included a sample analysis of the beginning of Schönberg’s Op. 11, No. 1. Included is an excerpt of a query to search the first six measures of a VirtualScore of Op. 11 for occurrences of the interval-class vector (1,0,1,1,0,0), corresponding to the pitch-class set (0,1,4). As this composition is literally a textbook example of this type of compositional pitch structure, it was chosen for its clear demonstration of the findings of my software.

After processing for less than a half-second, my software located 46 occurrences of the (1,0,1,1,0,0) interval-class vector within the first 6 measures of Op. 11. As space does not permit a complete listing, a sample is included below. Output is in the following format:

Note: PC is used as an abbreviation for pitch-class. Underlined values are replaced with specific data according to the following specifications:

- **PCX:** Used as a placeholder for the PC of interest
- **PCX-rhythm:** placeholder for the rhythm of the note sounding the pitch-class PCX
- **PCX-location:** placeholder for the location (in the form measure: beat, sub-beat) of the note sounding at the pitch-class PCX
- **IV:** Interval-class vector of the pitch-class set of interest

$$(PC_1, PC_2, PC_3) \ (IV)$$

$PC_1 \ PC_1$-rhythm at $PC_1$-location, $PC_2 \ PC_2$-rhythm at $PC_2$-location, $PC_3 \ PC_3$-rhythm at $PC_3$-location

Example:
The following is a specific instance of located information of interest, where

- $PC_1=B_3, PC_2=Ab_3, PC_3=G_3$
- $IV=\{1, 0, 1, 1, 0, 0\}$

Measures 1-5 of Arnold Schönberg No 1 from “3 piano pieces, op. 11” /UE 2991
The results of searching the entire composition are too extensive to list, as my software was able to locate and list 8,694 occurrences of the (1,0,1,1,0,0) interval-class vector when restricted to note groupings contained within three beats.

Such an analysis would have taken an extraordinary amount of time to perform by hand. By using computers to handle repetitive tasks quickly and efficiently, musicians are able to devote more time to forming musical opinions as to the meaning of such findings. In the above example, interested musicians can immediately begin examining computer identified note groupings forming each occurrence of the desired interval-class vector, and incorporating these findings into their analyses (or interpretations) without first spending hours locating such information by hand. The software is able to rapidly assist with the first steps of an analysis, and allows musicians to concentrate more intently on what matters – interpreting the music.
6) Future Work

Although for purposes of this project my software is limited to examining pitch structure, the VirtualScore and accompanying analysis software were designed to allow for automation of any analytical task. To this end, in addition to pitch information, the VirtualScore contains mechanisms for analyzing note locations, rhythms, dynamics, and instrumentation.

The analysis software contains the flexibility to analyze according to any user-defined method. Currently, it is limited to making examinations based only on interval-class vectors, however in the future this can be extended to make examinations by other means of comparison. Taking advantage of the VirtualScore and accompanying analysis software components already created for this project, other types of analyses (e.g. comparisons of rhythm, duration, or harmony) require only defining new metrics for evaluation.

Rather than performing a limited search for a pre-specified grouping of notes, in the future it may be possible to examine an entire composition for important information not identified outright by the user. By generating an extensive database of information describing pitch structure, it will be possible to detail the frequency of every interval-class vector throughout an entire symphony. Since it will be possible to examine, mathematically, the prominence of a particular pitch grouping in relation to other groupings, it will be possible to better understand the role of pitch structure in comprehension/appreciation of a given composition.

Moving beyond the concept of selecting and then examining individual note groupings, in the future VirtualScore technology can be adapted to other analytical tasks. For instance, it will be possible to generate contour graphs, or even other graphic representations of music. In fact, many contemporary analytical techniques are mathematically oriented and would lend themselves well to automation.

Examples include identifying pitches defining tone rows, locating subsets and supersets of a given set, and even performing statistical examinations of pitch structure to determine potential key areas.

Alternatively, the sheer speed of computer automation opens up new doors for large-scale, brute-force experiments. Given a series of VirtualScores, it would be possible to scan all for a given musical query. For example, the entire compositional output of a composer could be searched for a particular pitch/rhythm/instrument grouping, with a detailed summary shedding insight on compositional techniques used by particular composers. By carefully selecting the series of VirtualScores scanned, theorists can begin to make comparisons of changes in a composer’s compositional style throughout his or her career.

Despite the plethora of musical applications for VirtualScore technology, the most exciting prospects are those that are yet to be discovered. The flexible nature of the VirtualScore and accompanying analytical software allows for adaptations to future analytical methods yet to be defined. The architecture and design of the system will allow users to create new analysis metrics, and after defining only a minimal amount of attributes, re-search previously examined VirtualScores in a new manner.

Much as a meteorologist must combine raw data describing weather conditions with his or her own background and prior knowledge to construe an accurate weather forecast, it is ultimately up to the musician to take the raw data generated by my software and use it for a musical end. With automated assistance it will now be possible to instantaneously explore the prominence of note groupings, rapidly examine a composer’s choice of instrumentation, ranges, dynamics and/or note density, or promptly investigate hundreds of questions triggered by examining an automated analysis’s results. What my software offers is not an ultimate answer to, but the ability to explore, virtually any musical question.
7) Conclusion

In working on this project, my goal was to lay the necessary foundation for future work in automating the analytical process. Having completed the design of the VirtualScore and implemented one method of using it for analysis, I find myself ready to begin the exciting process of designing software to perform more sophisticated analyses of music. Using the VirtualScore in conjunction with future software components designed to take advantage of its strengths offers enormous potential for creating programs to automate various analytical tasks. The conclusion of this project represents the beginning of the next stage in exploring the tremendous possibilities afforded by utilizing the speed and power of computers to further our musical understanding.

8) Appendix 1: Illustration of SMF’s shortcomings

Below is a simple example of one of the SMF’s critical shortcomings - difficulty in ascertaining a note’s length. In order to determine this, both note-on and note-off messages must be identified and paired together. These messages are not necessarily located within close proximity of one another. For example, even in the simple, 184 byte SMF representation of the musical example below, the bolded messages (defining the G in the lower staff occurring on beat one, measure one) occur 26 bytes apart, (at bytes 129 and 155, respectively).

Sample musical excerpt:

```
    
```

Corresponding SMF representation of this excerpt:

```
77 86 104 109 0 0 0 5 0 1 (0)
0 3 4 0 77 86 114 107 0 0 (1)
0 27 0 255 84 4 7 24 6 2 (2)
0 255 89 2 0 0 253 81 3 5 (3)
9 137 104 132 224 0 255 47 0 77 (4)
84 134 107 0 0 0 43 0 255 3 (5)
11 71 114 97 110 100 32 80 105 97 (6)
120 113 0 192 0 0 144 65 126 (7)
0 128 65 0 152 0 144 67 66 116 (8)
0 128 67 0 132 184 0 255 47 0 (9)
77 86 134 109 0 0 0 77 0 255 (10)
3 13 73 97 109 109 111 110 100 32 (11)
79 114 103 97 110 0 166 16 0 148 (12) // Note-On, G3 channel 4
67 64 0 148 71 64 0 148 72 64 (13) // (Note Index 67 ~ G3)
0 148 74 64 160 0 132 71 0 0 (14) // 4096 pulses
148 71 64 136 0 132 67 0 0 132 (15) // 1024 pulses, Note-off
71 0 0 132 72 0 0 132 74 0 (16)
0 148 72 64 136 0 132 72 0 132 (17)
```
While it is therefore possible to determine a note’s length by summing the elapsed time values between note-on and off messages, the complexities and time involved in performing this operation serve as an enormous hindrance when determining the lengths and in turn the rhythms of thousands of notes. Obviously, this is unacceptable when performing any type of exhaustive, in-depth analysis.

The 23 bytes in between the note-on and note-off messages contain the beginnings of several other notes, as well as several timing messages. These timing messages (underlined for clarity) indirectly define the length of the note. By decoding each in turn and adding them together, it can be determined that 4096 + 1024 = 5120 pulses occur between the note’s beginning and end. Comparing this against the PPQN (Pulses Per Quarter Note) setting of 1024, the note is shown to be precisely 5 beats long.

8) Appendix 2: Glossary of Terminology:

- The following glossaries are intended only as a cursory overview, offering brief explanations of key terms used throughout this article. It is NOT intended as an overview of 20th Century theory, but rather a simple review of basic vocabulary used in this article.

- Terms appearing elsewhere within the same glossary are italicized for convenience and clarity.

Interval Class (IC): The post-tonal equivalent of tonal intervals (e.g. Major 2\textsuperscript{nd}, Perfect 5\textsuperscript{th}, etc). An IC represents the number of half steps between two pcs. For pcs more than 6 half steps apart, the complement of the interval is used, (the complement of an IC is defined as being 12 minus that IC).

ICs correlate with tonal intervals in the following manner:

<table>
<thead>
<tr>
<th>IC</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC 1</td>
<td>minor second, Major seventh (m2, M7)</td>
</tr>
<tr>
<td>IC 2</td>
<td>Major second, minor seventh (M2, m7)</td>
</tr>
<tr>
<td>IC 3</td>
<td>minor third, Major sixth (m2, M6)</td>
</tr>
<tr>
<td>IC 4</td>
<td>Major third, minor sixth (M2, m6)</td>
</tr>
<tr>
<td>IC 5</td>
<td>Perfect 4\textsuperscript{th}, Perfect 5\textsuperscript{th} (P4, P5)</td>
</tr>
<tr>
<td>IC 6</td>
<td>Augmented fourth / Diminished 5th (A4, D5)</td>
</tr>
</tbody>
</table>

Interval-class vector: A way of expressing the interval content of a pc-set. An interval-class vector is a summary all ICs occurring in a given pc-set. For example, the interval-class vector of the pc-set (0,1,4) contains three intervals, IC 1 (m2), IC 3 (m3), and IC 4 (M3). Therefore, the interval-class vector representing the pc-set (0,1,4) is (1,0,1,1,0,0).

Java: The programming language chosen for implementation of this project. One of Java’s strengths is its platform independence, meaning it can run on virtually any operating system (including both Windows and Macintosh).

MIDI Message: A grouping of data (typically 2-3 bytes) that signals some event. Examples of MIDI messages include: note-on, note-off, program change, pitch-bend, etc. Only note-on and note-off messages will be discussed in this paper, as they define the characteristics of a particular musical note.

Normal Form: The most compressed way of expressing the pcs in a given pc-set. As an example, the normal form of the group pcs (G, A#, B) is (0,3,4). Note that normal form is not the same as prime form, although pc-sets sharing a normal form will also have the same prime form.

Object: A virtual representation of any real world “thing.” The term object refers to a method of storing data as well as offering the ability to manipulate that data. The VirtualScore is an object, as are all the notes contained within it.

Pitch-class (pc): When examining post-tonal music, it is often helpful to consider occurrences of a note in separate octaves as equivalent, in which case the octave designation of a pitch is no longer significant. The pitch “C3” (middle C) is a member of the pc “C” (likewise, the pitch “D#5” is a member of the pc “D#”). Enharmonic equivalence is also assumed, and therefore C# and Db are considered to be the same pc. Given both octave and enharmonic equivalences, the notes C#2 and Db 5 refer to the same pc of C# (or Db).

For greater clarity, and to resolve the issue of equivalence between C# and Db, pcs (pitch-classes), are often referred to by integers values, rather than letter names. Pcs are mapped to integer values according to the following table:

<table>
<thead>
<tr>
<th>C</th>
<th>0</th>
<th>F#, Gb</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>C#</td>
<td>1</td>
<td>G</td>
<td>7</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>G#, Ab</td>
<td>8</td>
</tr>
<tr>
<td>D#</td>
<td>3</td>
<td>A</td>
<td>9</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>A#, Bb</td>
<td>10</td>
</tr>
<tr>
<td>F</td>
<td>5</td>
<td>B</td>
<td>11</td>
</tr>
</tbody>
</table>

Pitch-class set (pc-set): Pcs can be grouped into sets, containing up to twelve 12 pcs. Pc-sets contain no more than one occurrence of any given pc. For example, the pc-set of the group of pcs (G, G#, B) is (7, 8, 11).

Prime Form: A special case of the normal form of a pcs. Prime form is simply the transformation of a given normal form that is considered to be most compact. For example, the group of pcs (G, A#, B) has a normal form (0,3,4), and a prime form (0,1,4). Both (0,1,4) and (0,3,4) are inversional equivalents of each other, but (0,1,4) is considered more tightly packed to the left, and therefore the prime form of both pc-sets. For a more detailed explanation of inversional equivalence or any other terms pertaining to 20th Century analysis, the interested reader is referred to Joseph Straus’s Introduction to Post-tonal Theory.

Standard MIDI File (SMF): An extension of MIDI that allows for MIDI events to occur at specific times. Originally designed to facilitate inter-sequencer communication, the SMF format is the current standard for digitally representing and communicating music.

VirtualScore: A new data structure developed for representing music. Designed specifically for automating analysis, the VirtualScore is able to handle the six main requirements for analysis detailed in section 3.2.

NOTES

1 For instance, two adjacent notes could be represented in different chunks. As long as their delta time values dictate they be played sequentially, and they share the same channel number (and assuming no program changes in between the notes), it would be possible for two musically adjacent notes to reside physically quite far from each other.
Marc Chagall:
An Iconological Approach to Paris through the Window and Bride and Groom with the Eiffel Tower
by Rebecca Suzanne Roe

The artwork of Marc Chagall, infused with fanciful imagery and vivid colors, has mystified art lovers and critics since the early twentieth century. His paintings transport viewers into an extraordinary dream world, filled with flying cows, disoriented drunks, and lovers who can defy gravity. The creatures and images are so fantastic that viewers are often left with feelings of confusion and awe. Chagall’s artworks allow viewers to reclaim their childhood, when their imagination was their greatest toy. Nevertheless, observers of Chagall’s art, particularly the more mature, long for a rational explanation of his forms. Viewers want to know exactly what the painting means and what the artist intended. Although Chagall himself denounced the individual interpretation of his symbols,1 I believe that the recurring images and themes evident throughout Chagall’s portfolio lend themselves to an interpretation.2 In conjunction with aspects of his biography, an analysis of these images and themes help to form a reading of Chagall’s paintings Paris through the Window (1913) and The Bride and Groom with the Eiffel Tower (1928), two of Chagall’s most puzzling works. The conflicts of love, home, and career that are brought to life in Paris through the Window appear to be subsequently resolved in Chagall’s later work, The Bride and Groom with the Eiffel Tower.

Chagall’s personal background, notably his Russian and Jewish heritage, greatly influenced his artwork. He was born on July 6, 1887.

Rebecca Suzanne Roe received an honors bachelor’s degree in English and in art history, with a minor in studio art, from Saint Louis University in 2002. She is currently pursuing a master’s degree in art history at the University of Missouri-Columbia.