OMDN at the MediaEval 2014 C@merata Task: Using CPNView to Answer Questions about Scores

Donncha S. Ó Maidín
University of Limerick
Department of Computer Science and Information Systems
Limerick, Ireland
Donncha.OMaidin@ul.ie

ABSTRACT
This paper outlines the participation of OMDN at the C@merata task. It is based on the CPNView container-iterator model and associated tools developed in earlier work.

1. The Score
Unlike a natural language document, a music score cannot be represented conveniently as a string of around seventy characters. While natural language may be represented effectively using a linear array, a data structure that captures the relationships between entities in a music score does not readily fit into such, or into any of the simpler computing data structures, such as arrays, lists or trees. While all of the relationships between score entities might be successfully captured within a graph structure, the resulting data structure is likely to prove too unwieldy to be of direct use for music information retrieval and analysis. The solution employed here is one of encapsulating this complexity using the familiar container-iterator approach. This enables the underlying complexity to be hidden. The aim is to provide an abstract view of the score through a convenient set of member functions. These member functions are designed to facilitate access to score data in ways that parallel how we might interact with the score document, while keeping the underlying representational complexity secret.

It is worth focusing firstly on the task of representing a music score in a file. The only requirement is to have a convention that represents each visible item as it exists in the score together with its contiguity details. Many codes exist for doing this. Codes that are employed for transfer of scores between different notation programs, if they yield consistent results, must capture the basic information content of a score. Codes designed for other purposes, such as MIDI code that was designed for capturing selected gestural information of a keyboard player are unsuitable for score representation.

Scholarly research should always be based on value-neutral representations that accurately model the symbols on the score and only those symbols. Notionally, the adequacy of a representation might be tested if a number of engravers succeed in producing musically equivalent versions from the representation. Additionally the representation should have a one-to-one correspondence between encoded entities and visual entities, stripped of layout details. In practice value-neutral representations are rare, but this need not always be a hindrance to scholarly research where adequate filtering is possible. Strict observance of these representational norms is the only way to guarantee that algorithms will work properly when applied to any score. Algorithms that depend on any unnecessary additions will work only with files that contain such unnecessary additions. This will be the case irrespective of the tools used for the analysis, including Humdrum, Music21 and CPNView.

1.1 Observations on the Questions
“A short noun phrase in English referring to musical features in a score”, involved the specification of time values, pitches or intervals, in some cases joined by “followed”, “followed by”, “then”, and in other cases qualified by location, such as “bass”, “bass clef”, “in the voice”, “in the left hand”, “on the word”. Oddly, none of the test questions referred to strong or weak beat transitions or to positions with a bar.

While parsing short noun phrases as used in this challenge, is not a particularly difficult task, solving some of the questions can lead to difficult music challenges.

Of the more musically challenging questions are “melody with accompaniment” (It would be interesting to run solutions to this on all the scores in the challenge).

At a more complex level the questions that involve distinguishing between homophony, monophony and polyphony can raise complex issues.

One such is where monophony has implied harmonies or counterpoint. A score that may be judged monophonic on the basis of having only one note playing at a time for most of its duration. However a reasonable argument might be made that the same score has an underlying harmonic texture. Interestingly a perfect cadence was sought in one of the questions on the Bach 'Cello Suite, where the V chord is implied in the melodic line.

In cases where a distinction is being made between homophony and counterpoint, it would be interesting to see if lines in substantial parts of the scores follow the rules of first species counterpoint.

A number of questions imply a knowledge of key such as seeking Ia triad, lb triad and perfect cadence. Unfortunately all of the five questions for identifying perfect cadences had 'mode' tags present. Such tags have no corresponding entry in the score are foreign to a value-neutral score encoding. In practice tags such as this have have a tendency to be error-prone, as in the case in the training sample for “Sing we and chant it”. In any case, reliance on such tags replaces part of a music retrieval, with a text retrieval task and avoids doing music analysis. Additionally any solution reached in this way will not work reliably on other scores where this unnecessary tag is absent.

A successful identification of a V-I chord sequence at the end of the score can be relatively simple, although where suspensions and retardations are present it can be a bit more challenging.

A more complex challenge is to identify any intermediate perfect cadences. Here in addition to identifying V-I chord transition various other factors will have to be considered, as a V-I progression is a necessary but not a sufficient condition for a perfect cadence.
1.2 Model and Algorithms

A human reader may wish to access elements in a monophonic score from some starting point, such as from the score beginning or from the start of a section or even from some arbitrary point. Adjoining elements might then be visited in time sequence. Traversing polyphonic or homophonic textures may give rise to more diverse approaches to traversing the score. Any information relating to a score element is available to the human reader when the element is visited. In the case of a note, this includes pitch information, source of any accidental alteration, duration information in various formats and contextual information such as key signature, time signature, clef and position within the bar is available.

Such considerations led to the development of a container-iterator model to facilitate access to a score in ways that parallel how a human might perform the similar actions.

Common Practice Notation View, or CPNView was used to answer a subset of the questions in the C@merata challenge[1].

CPNView formed the main topic of a PhD dissertation[2]. The name CPNView was not used in the dissertation, but appeared in later publications[3][4][5].

CPNView is a very basic tool that focuses on score representation and not solving any particular set of problems. It models a score as an object-oriented container in a manner similar to that used for other data structures found in computer science textbooks. The CPNView model is designed to provide a value-neutral and objective representation of a score from common-practice notation. The score's internal content is available mainly using iterators. The iterator object keeps track of the context in which an object resides in addition to providing access to the score object itself through its member functions.

The iterators and their member functions can be viewed as paralleling the actions of a human reader. Typically, a human might access a score from the start and read through it serially. For some purposes the reading may traverse the score along one of the staves. Where a harmonic or polyphonic texture is of interest it will be desirable to access it as a sequence of vertical slices in time order.

2. CPNView

2.1 Score Object

The meaning of symbols in a score depends on their preceding context. For example a note's stress and pitch are determined by various scoping mechanisms. The emphasis a performer places on a note is influenced by its position in the bar and by the time signature. Such contextual effects may be modified by an attached symbol such as marcato. In a similar way the pitch of a note is determined by its vertical position, by the nearest preceding clef sign and key signature, and possibly by a preceding note within the bar at the same pitch level, and finally by an accidental placed on it. In CPNView, such contextual information is resolved automatically and the context is always available through iterator member functions. In this way the user is freed from the need to keep track of such scoping concerns, concerns that would otherwise be substantial barriers to development.

CPNView models a score as an object-oriented container in a manner similar to that used for other data structures such as queues, lists and graphs. The model is designed to provide a value-neutral objective representation of a score from common-practice notation. The score's internal content is available using iterators. The iterator object, through its member functions, in addition to providing access to the score object itself, also keeps track of the context in which the object resides. The iterators and their member functions can be viewed as paralleling the actions of a human reader.

In CPNView the score object may be created by specifying a file path.

Score(score(path));

This model requires no user knowledge of how the score is represented in a file. Currently, reasonably complete components are available for importing *kern [7], EsAC [8], ALMA [9] and a subset of MIDI. An incomplete MusicXML component was developed for the current exercise.

Access to all of the internals of the score is facilitated through an iterator object thus:

ScoreIterator cursor(score);

Here an iterator is created. It has the arbitrary name "cursor", "cursor" points to the start, and can be used to visit all of the objects in a score. Where the score contains multiple staves, this is an appropriate iterator for harmonic analysis. Here the score iterator will iterate through all of the entities in the score in time sequence. In traversing notes and rests, movement is along a time slice from the start of the uppermost note or rest on the highest stave to the lowermost note or rest on the lowest stave. Having completed the time slice, the iterator repeats this movement for the next time slice. The next time slice occurs where there is any point of interest, such as any note or rest transition. The 'point of interest' is a concept that was introduced by my Michael Kassler [11].

ScoreIterator cursor(score, 1);

The second iterator form above has an additional parameter and is used to iterate a single staff, staff number 1 in this case.

The function to step through all of the objects in the score is the step member function. The step function returns a value true as long as a succeeding object exists. The following code skeleton can be used to visit all the score objects, and values and contents become available to any code that replaces the ellipsis.

while (cursor.step()) {...}

If it is required to visit only the notes in the score, a parameter may be given to the step function. The following code fragment will count the number of notes in the score.

long count = 0;
while (cursor.step(NOTE)) {count++;}

All of the D notes in a score will be visited and printed by

while(cursor.step(NOTE))
{
  if (cursor.getAlpha() == 'D')
  cout << cursor << "in";
}

Were a human reader to start at some intermediate position in a score, it would become necessary to establish the context by tracing backwards. An alternate to using the score iterator's step
function from the start of the score is to arrive at an arbitrary location in the score. This is provided by the locate function.

cursor.locate(3, Rat(1, 4));

The last function positions the iterator, named “cursor”, a crotchet or quarter note distance into bar 3. Again all pitch and
durational information is retrievable together with all scoping
information.
The ScoreIterator object has a comprehensive range of member
functions to retrieve all of the information that is contained
within the score. It is the responsibility of the iterator to resolve
all of the contextual information. Such contextual, or scoping
information includes resolving the exact pitch of a note. Other
contextual or scoping mechanisms are involved in retrieving
active time signatures, clefs and within-bar placements. All of
these are automatically resolved. This mechanism proved useful
in building a component that generates the XML answers.
The following code was used to generate the XML answers from
a start and end location (cursorA and cursorB), using the
specified division values (divs1 and divs2). The function named
startDivs does the calculations on the durations (expressed
conventionally as rational numbers in CPNView) to produce the
required start offset value for insertion into the XML text. The
rational numbers in this function returned by the function
generateStartDivs(). Rational numbers have members n and d, for
access to the numerator and denominator values. A similar
function named endDivs does the corresponding calculation for
end offsets.

String XMLPassage(ScoreIterator & cursorA, long divs1,
                   ScoreIterator & cursorB, long divs2)
{
    String answer;
    answer += Strings(endDivs(cursorB, divs2));
    answer += Strings("n" / > n");
    return answer;
}

long startDivs(ScoreIterator & cursor, long divs)
{
    return (cursor.getBarDist().n * 4 * divs)/
            cursor.getBarDivisions().div + 1;
}

code produces output such as
    <passage start_beats="4" start_beat_type="4"
            end_beats="4" end_beat_type="4"
            start_divisions="1" end_divisions="1"
            start_bar="8" start_offset="1"
            end_bar="8" end_offset="3" />

A natural language query that searches for all of the D notes and prints the answers in the required XML format, assume a required division value of 4 is achieved by

    while (cursor.step(NOTE))
        if (cursor.getAlpha() == 'D')
            cout <<
            XMLPassage(cursor, 4, cursor, 4);  (1)

The version of this code used in this exercise is different
from the above in that (1) the output is made to a list of strings
instead of printing the result, and (2) the "if (cursor.getAlpha() == 'D')" code is replaced by an extensive pattern matching
function that selectively performs matching of accidentals in
effect, time values and note qualifiers such as fermata, staccato and marcato as well as pitch names.

In addition to modeling a score, CPNView has a set of
components that facilitate processing musical information. They
include List, Set objects, objects for cumulating statistics and an
object for calculating pitch class sets.

Of particular interest is the set object that can be used for
calculating prime forms for harmonies and modes. Prime form
names are based on the classification system of Alan Forte [10].
The original classification, that was created for the analysis of
atonal music is available with CPNView. For C@merata a
modified version of pitch class sets can be used. It is suitable for
the classifying of tonal pitch combinations. Pitch class sets have
been used for classification of scale and modes as well as for
harmony [3]. The value of this can be best illustrated by an
example. The first inversion of the chord of E minor might be
represented by a set of chromatic note values represented here
using MIDI note numbers { 67, 71 76 }, or an open version of
the same chord by { 43, 71, 100 }. These sets are converted into
pitch classes by a modulo 12 operation. Here both produce {7, 11, 4}. By rearranging the numbers in ascending order we get
{4, 7, 11}. Next we transpose them so that the first member is 0.
This gives the set {0, 3, 7}. The next step is to look at alternate
cyclic transpositions of these, and pick the one with the smallest
intervals from start to end. This is so as to seek a unique
configuration for any combination of pitches. The unique sets are
give identifying labels such as 3-11. We can skip this step here as the smallest possible interval is the minor third from 0 to
3. If we look up the table of Forte's prime forms for the pitch
class set, we find that it has the name 3-11. There is one
additional step that Forte used - that is one of inverting the
intervals – a step that brings tonally dissimilar chord into the same classification. This makes the basic pitch class classification unsuitable for tonal as opposed to atonal music. This can be illustrated by repeating the above exercise with E major instead of E minor. In this case we would have arrived at the same prime form in Forte’s classification. Clearly this is unsatisfactory for tonal harmony classification where a distinction between major and minor triads is important. However if we flag the prime forms to indicate whether or not an inversion operation was used, we get a satisfactory classification for chords. In addition to proving useful for harmonic analysis, the modified pitch class sets may be used for identifying modes and keys. The resulting forms are labelled in CPNView as 1-3-11 for a major triad and 3-11 for a minor triad. The additional 1- depends on whether or not the inversion operation was used to arrive at the name.

Sample code to classify and report on all of the vertical combinations of pitches in a score is illustrated the following 9 lines of code.

```cpp
Score score(filepath); //1
ScoreIterator cursor(s); //2
PitchClass pcs; //3
while ( cursor.step() ) //4
{
  if ( cursor.isA(NOTE)) //5
  {
    pcs.put( cursor ); //6
    if ( cursor.isSystemLowermost()) //7
    {
      cout << pcs.getName() << "n"; //8
      pcs = PitchClass(); //9
    }
  }
}
```

Line 1 loads the file located at filepath into the container score. Line 2 creates an iterator called “cursor” for traversing score in time-slice order. Line 3 creates a container “pcs” for holding a set object. Line 4 controls the loop that traverses all score entities. Line 5 selects only note object for processing. Line 6 puts that note object into the set pcs. Line 7 controls action if the cursor has reached the lowest object in the time-slice being visited. Line 8 outputs the set name.

At the end of vivaldi_concerto_rv299_largo.xml, gives the following pitch class sets for the last three time slices. The 4-note set between the two major triads is due to the additional E that anticipates the tonic.

![Image of pitch class sets](image)

Another potential example is where pitch class sets can be used to identify scales. The pitch class set 7-35 corresponds to the notes of the diatonic major scale. The following lines of code -

```cpp
Scoreiterator cursor(score);
PitchClass pcs;
while ( cursor.step() )
{
  if ( cursor.isA(NOTE))
  {
    pcs.put(cursor);
  }
  cout << pcs.getName() << "n";
}
```

When applied to purcell_fie_nay_priethee_zd10.xml, outputs the pitch class set for the 7-note major scale 7-35

While the pitch class object would have proved useful in the current exercise, it became impossible to meet the required deadline in time to use it. It has been deployed in the meantime and is included here to demonstrate how it could be used for harmonic and tonal analysis.

### 2.2 Language Processing and Search Construction

For the identification of single notes or rests, the text is parsed and the specified fields are inserted into a search template. This involved using elementary string processing to recognize the references to notes or rests in order to form a search template. Search is performed in the manner of (1) above. Where such elements were connected by such phrases as “followed by” or “then” the same recognition was performed repeatedly on advancing the score iterator by one note or rest as required.

### 3. Evaluation

Participating in C@merata involved developing a software component for importing MusicXML files into CPNView. This is a non-trivial task that did not fit the available completion time. Much of the time during the three-day limit for completing the assignment was taken up with getting a very basic subset of MusicXML working. As a result only a tiny subset of the task could be completed. The main lacunae arose from (1) files that had multiple simultaneous notes or the same staff were not processed, (2) queries were limited to identification of notes or rests; questions about intervals and other features including harmonic aspects, were not attempted. (3) there was not time to check the accuracy of the results and some major errors and misinterpretations in the submitted answers. These are documented in (a) to (d) below.

(a) when a time value such as a minim was specified, searches were performed for minim-valued notes and rests instead of for notes only. This resulted in getting all the correct answers for notes, but a lot of incorrect answers where minim rests occurred. However, notes and rests with the time values were correctly identified.

(b) in identifying pitches, the natural language component searched for notes one octave higher than was required. This produced the odd answer that was coincidentally correct. This was due to lack of definition in the answers where the exact
location of the target in relation to staffs or parts was not captured. Visual inspection of the scores showed that the submitted answers correctly identified and reported all occurrences when the calibration error was taken into account.

c) the MusicXML bar numbers were not imported, but CPNView allocated bar numbers, starting with bar number 1 for the first complete bar. While this is the recommended norm for encoding MusicXML, it was not adhered to in all of the scores in the exercise. A visual check confirms that correct answers were produced when due allowance was made for this discrepancy.

d) where there are multiple answers to the same query, for example in a search for 'E' notes, multiple 'E's might occur at the same place and with the same duration. Repeated instances, one for each occurrence, were included in the submitted answers, but not recognised as valid. Here the answering system did not allow for the specification of the exact vertical placement involved in answers.

3.1 CONCLUSION

3.2 Considerations for Score Encoding

(1) The practice of placing exact pitch information on each note, while making the analysis task easier, is an additional entry that is not required in a neutral representation of the score. The exact pitch of a note is determined in the first place by an attached accidental sign, secondly, in the absence of an attached accidental, by any accidental alteration to another preceding note at the same pitch and within the same bar, and finally, if neither of these mechanisms are present, from the key signature. Unnecessary markings are added, presumably for convenience, violate the integrity of the representation and may lead to having inconsistencies. This practice is not inherent to MusicXML, and may well be the cause of an error on Page 5 of the camerata_2014_task_description_v7 document, where an extract from J.S. Bach's, Das Wohltemperierte Klavier, Book 1, Prelude No. 2 in C minor BWV 847 is used. Here some of the E naturals were missed in the sample answers provided. If these results were produced by software, it is likely that the XML encoding truly represented the score, but the software failed to keep track of the context of the three following E naturals.

(2) Additional information of an interpretative nature is provided in many cases that adds new information to the score. One example is where the key mode is specified.

3.3 Reflections on Answering

The method of specifying locations is problematic because

(1) Locations are not fully specified. They do not indicate the staff or part on which an object is placed. While the time point in the score is represented, specification of the line/part/stave of the line is absent. All of the scores contain at least one part. Parts in the set of scores contain zero, one, or many staffs. A scheme such as the following might be adopted and incorporated into the specification.

- \( P1 \) to specify part 1 where the part contains no staff specification.
- \( P1.2 \) to specify staff 2 in part 1.

(2) Furthermore in the case of chords or polyphony on the same staff, the location of a note or rest is not accurately identified. In such cases if notes or rests are numbered 1, 2, 3, ... from highest to lowest, then an additional specification can be added, such as \( P1(1) \) or \( P1:2(1) \).

The B notes on the final chord in the above example could be distinguished as \( P1(2) \) and \( P1(4) \), assuming that belongs to Part P1. If the line was otherwise encoded as stave 1 of part P2, these might be referred to as \( P2:1(1) \) and \( P2:1(2) \).

(3) The system of beats and divisions is not intuitive, and becomes difficult in cases where passages containing multiple groupettes, such as where a triplet and quintuplet occur in the same bar. While any system will bring its own complexities, it would be best to rely on rational values since these are already integral to the notation and embedded in the way musicians think about time values. The time signature 4/4 represents four quarter notes. A sequence of 4 crochets/quarter notes in 4/4 time will have starting or end points at one of the following places: 0/4, 1/4, 2/4, 3/4 and 4/4, in a case where the bar contains a triplet for the duration of the first half of the bar, that is for a duration of 2/4 or 1/2, we just divide these by three giving 2/12 or 1/6. The resulting starting or end points of the notes can be specified at 0/6, 1/6, 2/6 and 3/6 (or 2/4). No decision is required to adequately represent the bar as is the case when beats and divisions are used. Rational numbers have multiple forms, 4/4 = 2/2 = 1/1. It is desirable in order to improve readability, to preserve, as far as possible the denominator from the time signature, although all equivalent values must be taken as correct.

An example of specification for the highest note \{1\} on Part 1 \{P1\}, stave 1 \{:1\}, start of bar 1 \{1(0/4)\} to end of bar 2 \{2(4/4)\}:

\[ P1:1(1),1(0/4);2(4/4) \]

A corresponding XML specification is

\[
<\text{passage part="1" staff="1" vertical_sequence="1" start_bar="1" start_position="0/4" end_bar="2" end_position="4/4"} />
\]

3.4 Comments on Task Description V7

Page 5: J.S. Bach, Das Wohltemperierte Klavier, Book 1, Prelude No. 2 in C minor BWV 847 Second sample question/answer:

Q: interval of a melodic sixth (any kind of sixth allowed)

---

Q: semiquaver E natural / sixteenth note E natural A:

\([4/4,4,2:3-2:3] \]

---

Q:.

---
harmonic intervals may occur across staves."

Query: Why were the following excluded from the answer?


D. Scarlatti, Keyboard Sonata in D major, K 430

Query: What were the following excluded from the answer?


Right hand

Query: Why were the following excluded from the answer?

D. Scarlatti, Keyboard Sonata in D major, K 430

Query: Second bar has Eb and Bb octaves and final bar has
Bb and F octaves. I don't think that it makes much musical
sense to exclude these from the answer. Is it because they
are written on different staves or is there some other reason
to exclude them? Page 4 of the Task Description states "On
the other hand, harmonic intervals may occur across
staves."

D. Scarlatti, Keyboard Sonata in D minor, K 1
Q: third

Query: The following are omitted. Intervals are harmonic
and lie across staves as in the previous case:

<table>
<thead>
<tr>
<th>Bar</th>
<th>Left Hand</th>
<th>Right Hand</th>
</tr>
</thead>
<tbody>
<tr>
<td>4/4,4,17:1-17:1</td>
<td>G</td>
<td>Bb</td>
</tr>
<tr>
<td>4/4,4,17:1-17:3</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>4/4,4,17:5-17:5</td>
<td>Bb</td>
<td>D</td>
</tr>
<tr>
<td>4/4,4,17:7-17:7</td>
<td>C</td>
<td>E</td>
</tr>
<tr>
<td>4/4,4,17:9-17:9</td>
<td>D</td>
<td>F</td>
</tr>
<tr>
<td>4/4,4,17:11-17:11</td>
<td>E</td>
<td>G</td>
</tr>
<tr>
<td>4/4,4,17:15-17:15</td>
<td>G</td>
<td>Bb</td>
</tr>
<tr>
<td>4/4,4,18:1-18:1</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>4/4,4,18:5-18:5</td>
<td>G</td>
<td>B</td>
</tr>
<tr>
<td>4/4,4,18:9-18:9</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>4/4,4,18:13-18:13</td>
<td>G</td>
<td>Bb</td>
</tr>
<tr>
<td>4/4,4,19:1-19:1</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>4/4,4,19:3-19:3</td>
<td>F</td>
<td>A</td>
</tr>
<tr>
<td>4/4,4,19:5-19:5</td>
<td>G</td>
<td>Bb</td>
</tr>
<tr>
<td>4/4,4,19:7-19:7</td>
<td>E</td>
<td>G</td>
</tr>
</tbody>
</table>

D. Scarlatti, Keyboard Sonata in D major, K 430
Q: harmonic 5th followed by harmonic 4th

Query: According to the earlier text "...On the other hand,
harmonic intervals may occur across staves."

[3/8,2,55:3-56:1] 5th from A in LH to E in RH, is
followed by A and D in LH.

There is an example in the fifth file in the XML code of the
training set where the dubious practice of the <mode>
annotation is used in the file for “Sing we and chant it”. This is
annotated as being in the major mode with no accidentals,
presumably C major. The first section of this piece has not a
single F natural in it, but has 5 F# notes. The middle section has
only F naturals, but the harmony suggests the mixolydian mode
on G. Finally F#s predominate with a final cadence ending on
chord G.

3.5 Comments on Evaluation Answers

Questions marked (*) reported all notes and included
different notes that had similar XML locations specified.

(*)Question No. 83: There are 32 quarter notes in this
piece, including 2 in the right hand. It appears that all of
the submitted answers are correct, with one entry for each note
identified.

(*)Question No. 92: There are 26 minims in total, including
3 in bar 15. As far as I can see all 26 are reported correctly.
There are 3 simultaneous ones in the final bar, and the solutions
contain one entry for each one of these.

Question No. 96: The following appear reported correctly
but not credited. These are in bar 6 bass (G3-G2), bars 6-7
bass(G2-A3), bar 9 bass (D3-C3),bars 9 & 10 bass (C3-B2), bar
10 bass (B2-B3),bars 10 & 11 bass (B3-E3), bar 13 (G3-G2)
Bar 15 was reported in error, as well as cases where the
minims were on different staves.

(*)Question 145: There are 4 dotted half notes in bar 2 and
four in bar 4. These appear to be reported correctly.

(*)Question 146: Five half note occurrences between bars
6, 8, 10 and two in 12, all appear to be reported correctly.

(*)Question 172: 11 instances in bars 1(1), 2(1), 3(1), 5(1),
7(1), 8(1), 10(1), 13(1), 16(3). all appear to be reported
correctly.

3.6 Recommendations for Future Trials

(1) All queries should be based on value-neutral
representation of music scores. Any additional information,
except perhaps for the most basic labeling of the initial text,
would best be removed and where this is not feasible, they
should not be used to arrive at solutions. Possibly participants
could opt to receive a set of score file that have unnecessary
entries filtered from them.

(2) The system of indicating locations in scores should be
revised so as to include details of line locations. Also, the
system of depending on beats and divisions should be
abandoned in favour of a more intuitive approach. The new
scheme should report multiple answers that share the same start
and end point.

(3) A uniform approach should be taken to bar numbering.

(4) Some deeper question might be distributed well in
advance of the scores, so that participants may have a chance to
refine their approach.

(5) The format of answering might be expanded beyond the
specification of ranges.
3.7 Suggestions for Future Tasks

(1) Refer to strong and weak beats in some questions. For example specify “rising perfect fourth weak beat to strong”.

(2) Seek rising scale passages of more than three notes.

(3) Seek repeated sequence of n (specify n) notes.

(4) Repeated sequence of n (specify n) notes that start on successive weak and strong beats.

(5) Seek melodic sequences of length n (specify n).

(6) Seek melodic palindromes.

(7) The format of questions and answers might be expanded to encompass forms for reporting single numeric or textual answers. This would facilitate questions involving estimations of mode and range.

(8) Find melodic range (involves having a structure for an integer answer).

(9) Identify key (involves having a structure for text answer).

(10) Identification of parallel simple and compound consecutive fifths and octaves.

(11) Selected questions might be run on all, or on a substantial subset of the scores in the set.

(12) Some music from later periods and other genres might be used.

4. REFERENCES


