OMDN at the C@merata 2016 Task: Approaches to and Issues Arising from Answering Natural Language Questions about Music Scores

Donncha S. Ó Maidín
Department of CSIS
University of Limerick
Limerick, Ireland
Donncha.OMaidin@ul.ie

ABSTRACT

The query is modeled as a parse tree where clauses from the query form children of the tree's root. Children of each clause, in turn represent the query's atomic units. A complete search involves visiting all score objects that are potential starting points for a match. Atomic units of the parse tree are compared, in sequence, with corresponding score objects. From each score object, appropriate paths are explored to find objects that match the sequence of tree's atomic units. The parse tree is traversed in a depth first order. Iteration paths through the score follow either vertical or horizontal routes, the former for chord or harmonic interval comparisons, and the latter for melodic comparisons. Iteration between clauses results in the selection of new score staring points, appropriate to the temporal relationship between clauses. Results were limited on account of problems with importing some of the MusicXML files.

1. INTRODUCTION

A natural language query is first split into component queries X and Y by scanning the query for one of the following structures:

- X followed by Y
- X against Y
- simultaneous X and Y
- X at the same time as Y
- X followed [distance] by Y
- X

For structures 1 and 5 queries X and Y are searched in sequence, while for structures 2, 3, and 4, X and Y are searched in parallel. Units within X or Y in turn result in sequential or parallel searches between their contained atomic units. A sequential search is involved where the atomic units in X or Y are notes of a melody or melodic intervals; a parallel search is required where X or Y contains notes of a chord or a harmonic interval.

Each atomic unit is represented by a template that contains details of all the necessary requirements for a unit match with a score object. An instance of an atomic unit may be a note, a rest or it may specify a more complex object or texture such as an arpeggio or scale. Templates also contain information on how they are linked together. Lists of templates are used to represent either sequential or parallel sets of notes or rests. Arpeggios or scales are examples of more complex atomic objects that are represented in single templates. In addition to carrying details of the target entity for matching, a template also carries relevant contextual information such as bar ranges, clef and instrument designations wherever appropriate.

The following examples use [>], [^], [] to represent respectively sequential [>], simultaneous [^] or terminal relationships in nodes.

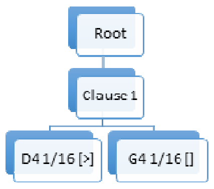

Figure 1. 'D4 G4 in sixteenth notes'

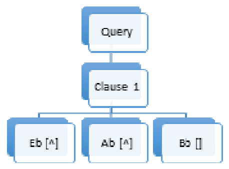

Figure 2. 'chord Eb Ab Bb'

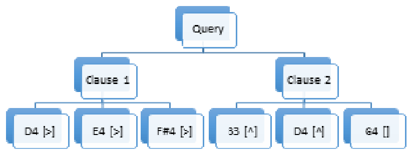

Figure 3.'D4 E4 F#4 followed by the chord B3 D4 G4

2. CPNView

Common Practice Notation View, or CPNView is used. CPNView formed the main topic of the author's PhD dissertation[1]. The name CPNView was not used in the dissertation, but appeared in later publications [2][3][4].

CPNView models a score as an objected-oriented container. The CPNView model is designed to provide a value-neutral representation of elements of the score notation. The score's internal content is available 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

MediaEval 2016 Workshop, October 20-21, 2016, Amsterdam.

as paralleling the actions of a human reader. Typically, a human might access a score from the start and read through it serially in time sequence. For some purposes the reader may traverse the score along one of the staves. Where a harmonic or polyphonic texture is of interest it is appropriate to access the score as a sequence of vertical slices, in time order.

A score object is created by specifying a file path:

Score score(path);

This model requires no user knowledge of how the score is represented. CPNView representations are built by importation from files in a number of different standard encodings.

Access to the internals of the score is facilitated by an iterator object:

ScoreIterator cursor(score);
or ScoreIterator cursor(score, 1);

The first instance creates an iterator that initially points to the start and can be used to visit all of the objects in a score in time order. This is an appropriate iterator for harmonic analysis. The second form has an additional parameter and is used to iterate a single stave; above the stave with identifier 1 is selected.

The iterator can step through all of the objects in the score using the step member function. The step function returns a value true following a test that the succeeding object exists. The following code skeleton makes all objects available, in sequence 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 as in the following code to count the notes in a score.

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

A locate member may be used to place the score iterator in an arbitrary position. For example the iterator may be positioned at the start of bar 20 by means of

cursor.locate(BARLINE, 200);

The ScoreIterator object has a comprehensive range of member functions to retrieve the information that is contained within the score.

A query that searches for all of the D notes and prints details of each note arrived at is achieved by

while (cursor.step(NOTE))
if (cursor.getAlpha() == 'D')
cout << cursor << "\n";

CPNView also has a set of components that facilitate processing musical information. They include container/iterator classes for Lists, Sets and Stores.

A specialised class exists for calculating pitch class sets. The pitch class object is based on the classification system of Alan Forte[6], that has been modified for the classifying tonal sets, such as those that occur in scales, modes and in harmony[2].

For processing each question, the initial phases consist of two separate steps: (1) the importation of the music file to build a CPNView representation of the score, and (2) the creation of a tree of matching templates from the query text. Following these input phases, the MusicXML file plays no further role. All subsequent processing is performed on the CPNView representation.

In the CPNView score representation, MusicXML layout and 'division' information is absent. XML fields for items such as 'mode' and 'voice' are discarded as well. Inclusion of such would undermine the ideal of a neutral score representation. Instead of relying on such added cues, the integrity of the analysis is best maintained by practices similar to those employed by musicians. For example voice leading, especially in keyboard scores, is best garnered from visual cues such as note stem directions and from knowledge of compositional principles. The challenge involved is to build software that mimics music expertise.

As outlined in the abstract and in the introduction above, a tree is built from the question text, with a template at each tree leaf that contains matching criteria for each atomic element of the search. An example is where a template is used for matching a note or a rest in the score. Such templates may additionally contain contextual information relating to factors such as bar range, clef and key signature identity. Note and rest templates may be grouped into larger clauses, corresponding to lists in the original query such as 'A4 C5 F4'. Such lists are used for both sequential and harmonic searches. Other types of query clauses contain single items such as Alberti basses, pedals, cadences, textures and also named chords corresponding to texts such as 'tonic' or 'IIa'. Each such clause is matched against a score by a specialized component.

A single query with search text 'A B' is performed as follows. For illustration purposes, the score is assumed to have only one stave containing a single melodic line:

1. Score is imported into CPNView.
2. A search tree is created with two template nodes, the first one for identifying the note 'A' and the second for 'B'.
3. A score iterator 'SI1' is created and positioned at the first note of the score.
4. 'SI2', a copy of 'SI1' is created.
5. If the note at 'SI2' is not 'A', move to (9).
6. Advance 'SI2' to the next note.
7. If the note 'SI2' is not 'B', move to (9).
8. Output the result.
9. Finish if no more notes are available, else increment 'SI1' to point to the next note and then go to (4).

The code that implements a generalized version of the above is more complex. It handles note lists of any size, handles search ranges corresponding to query text entries such as 'between bars 5 and 6' or 'in the soprano' and handles multiple staves and multiple sequential paths such as is found in keyboard music.

Components are available for the recognition of vertical combinations of notes and for other constructions such as those corresponding to texts such as 'major triad', 'dominant chord' and 'perfect cadence'. In some cases components may invoke other components. For example the component to identity cadences will also invoke key-finding as well as chord-identifying components.

While examples of coding that do substantial work are too long for inclusion here, the following is used to give a flavor of how CPNView programming works. It is based on one existing component used to check if the score iterator position in *cursor* is within the query template range. The template is called *templ*, and the range is between the values in *startBarNo* and *endBarNo*. The cursor object has a member function called *getCurrentBarNo()* that retrieves the relevant bar number for the iterator. Where no range is specified the template fields have a value of -1.

```
bool filterRange(ScoreIterator & cursor, Template templ)
{
   long scoreBarNo = cursor.getCurrentBarNo();
   if ( templ.startBarNo != -1 &&
        thisBarNo < nrst.startBarNo) return false;
   if ( nrst.endBarNo != -1 &&
        thisBarNo   > nrst.endBarNo)   return false;
   return true;
}
```

The existing function used is a bit more complicated in order to allow for variation in the ways that bar numbers get counted in the MusicXML scores used in C@merata.

All of the software is deterministic. Templates are formed from the queries and in cases where a query has not been envisaged beforehand no action takes place. The structure of queries is based largely on examples from the task description.

The execution of queries in some cases involves taking 'unsafe' short cuts that have worked with previous C@merata challenges. For instance the identification of melodic segments is done in a relatively unsophisticated way, by searching for unbroken sequences of notes of sufficient length, bounded is some way, such as by rests. This does not take into account the full range of possibilities for melody structuring. Also key identification is performed using the short cut of examining final chords and key signatures. One interesting challenge is presented in question1 59, 'passing modulation to B minor'. Answering this query requires a tonality-tracking component. Another challenging case can be found in question 12, 'imitation between Sopranos and Clarinet in measures 110-119', that requires the implementation of a melodic similarity algorithm.

3. MATCHING QUERY TO SCORES

The complexity of processing a search depends on both the score and on the nature of the query. An otherwise simple task applied to the score of a hymn tune might prove complex when applied to a piano score or to a score with transposing instruments. Some of the issues identified are listed below:

1. Query 'A3'. This specifies one of the simplest possible search tasks. However when applied to a score with transposing instruments, it yields two valid interpretations. The issue is whether the search is for a notated or for a sounding A3.

2. Query 'Bb4 G4'. This is a simple query when applied to a score with monophonic staves. Were it used to search a score with multiple simultaneous notes on some of the staves, an issue arises on whether or not voice-leading considerations are taken into account when identifying all Bb4 that are adjacent in time to G4. Where voice leading is not taken into account the query 'Bb4 G4' applied to Figure 1 below would yield [3/4,2,23:1-23:2]. The nub of the problem here is that much keyboard writing is based on historic practices of part writing from previous centuries. In the Scarlatti score fragment below, three parts or voices may be identified with pitch sequences (Bb4 C#5 D5 E5 F#5 G5) in the top part, (G3 B4 F4 E4 D4 C#4) in the middle part and (G3) in the lower part. Hence the answer [3/4,2,23:1-23:2] is incorrect.

3. A search in the fig.4 score for G3 B4 F4 E4 D4 C#4 should yield [3/4,2,23:1-23:6], although this solution is forbidden in the current C@merata rules that prohibit the identification of melodic sequences between notes on different staves, a rule that is reasonable, provided it is not applied to keyboard notation.

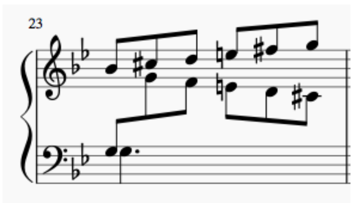

Figure 4. Scarlatti_k30 taken from C@merata 2015.

4. Search texts such as 'tonic triad' or 'triad of E major' where the component notes have a common starting and ending point, with no other pitch class present, are relatively easy to handle. However there are many cases where the identity of a chord can be maintained although notes foreign to it are present. Suspensions, retardations, passing notes either accented or unaccented, and pedals or inverted pedals are such cases. Context plays a major role here. Metrical considerations and the behavior of preceding or following notes have to be taken into account.

5. Keyboard scores present additional challenges for voice-leading and chord identification tasks. In many cases the music textures can be viewed as having an underlying traditional 4-part harmonic structure, when instrumental idioms such as octave doubling and the spreading out of chords and crossing of staves are allowed for. Figure 4 shows a case where the middle melodic line moves from bass to treble clef. Figure 5 illustrates that use of spread out chords that conform to good voice leading rules.

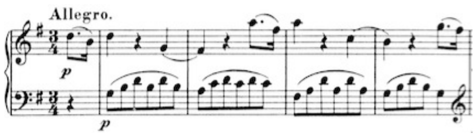

Figure 5. Mozart piano sonata no.5, first movement.

The melody has a chordal accompaniment with G major root position in bar 1, D dominant seventh second inversion in bar 2 continued into bar 3 but falling its first inversion and returning to G major in bar 4.

There is a clear bass line G in bar 1 A in bar 2, F# in bar 3 and G in bar 4.

Two middle parts can be identified, the upper one on D throughout, and the lower one with B in bar 1, C in bar 2, A in bar 3 and B in bar 4. All of these lines are well behaved, avoiding forbidden parallelisms. Similar considerations are relevant to instance Alberti basses and some arpeggios.

6. The C@merata exercise provides an opportunity for testing definitions of music terms. The question arises regarding the meaning of some music terms. A designer of recognition of software might start with a definition from a music dictionary. While a dictionary definition gives necessary

conditions, they frequently leave borderline cases unresolved, or hedge around a definition with words like ‘usually’ or ‘normally’. The definition of ‘chord’, for example may include statements such as that it ‘usually contains more than two notes’. This reveals that there is a level of complexity involved that cannot be put into a concise manner suitable for a dictionary definition. An exploration of the use of composition textbooks could prove beneficial.

One way of examining some of the issues here is to attempt to find how musicians conceptualise a term.

A short exploration of some of the issues involved is undertaken here in an attempt to design a recognizer for ‘arpeggio’, using instances from questions in C@merata 2015 and 2016.

One music dictionary definition of ‘arpeggio’ is:

“A chord ‘spread’ – i.e. the note heard one after the other from the bottom upwards, or sometimes from the top downwards.” [5].

Interestingly, the answers in the Gold Standard 2015 are in accordance with the dictionary definition.

While the definition seems to meet the necessary conditions for an arpeggio, the question arises whether the definition is sufficient. It depends on the meaning of ‘chord’ that itself can be somewhat problematic, allowing for the possibility of a two-note arpeggio. Additionally, if an arpeggio-like construct is part of a larger structure, such as a broken chord, can it still be regarded as an arpeggio?

In a brief exploration, responses were elucidated from five practicing musicians to extracts used in C@merata questions. All respondents professional musicians with keyboard competence and were mainly from the teaching community. Two have substantial composing experience. The respondents were not initially aware that arpeggio recognition was at issue.

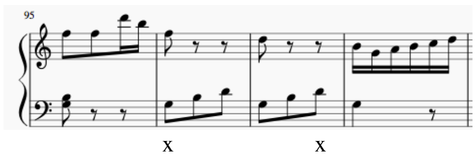

Figure 6. clementi_sonatina_op36_no1_v2

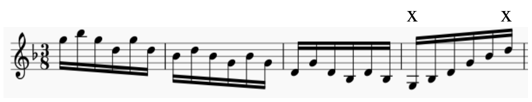

Figure 7. bach_violin_sonata_no1_bwv1001_presto

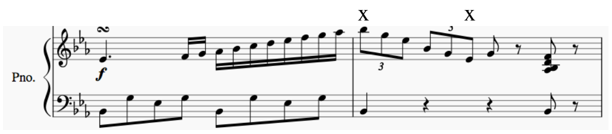

Figure 8. mozart_an_chloe_k526

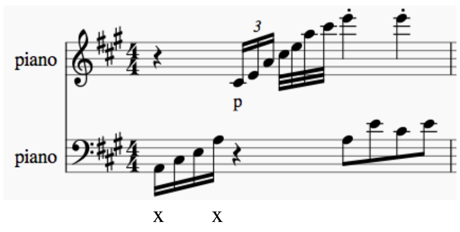

Figure 9. sonata02-4

Figures	Initial Responses ‘arpeggio’	Arpeggio Appropriate
4	20%	60%
5	20%	100%
6	80%	100%
7	80%	100%

Table 1. Participant responses to arpeggios in Figures 4-7

The majority of the respondents agreed with the designation except in the case of figure 6. Two of the participants conceptualized an arpeggio as having at least four notes, requiring one octave to be present. One found the term ‘broken chords’ to be more appropriate to the configuration in figure 6.

Following this two cases were presented where the arpeggio candidate notes lay in larger context. In the first case the candidate arpeggio was three notes within a melodic line; in the second case it lay in a larger broken chord context.

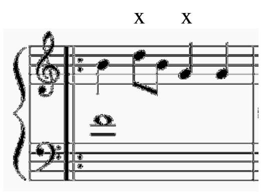

Figure 10. clementi_sonatina_op36_no1_v2

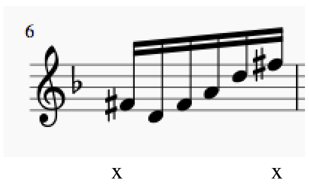

Figure 11. bach_violin_sonata_no1_bwv1001_presto

Figures	Agree Arpeggio Designation
10	40%
11	40%

Table 2. Participant responses to arpeggios in Figures 10-11

The opinion of the majority was not in agreement with the dictionary definition of ‘arpeggio’. Interesting both ‘yes’ respondents said that they did not make a great distinction between ‘broken chords’ and ‘arpeggios’, while the ‘no’ respondents did. Excluding the respondents that were ambivalent about this distinction would have resulted in both figures 10 and

11 from being excluded from the category ‘arpeggio’ by all of remaining respondents.

This exercise highlights a problem in the creation of recognition algorithms. It would make for an interesting investigation to test instances of performed music for the identification of features such as arpeggios.

4. FUTURE CHALLENGES FOR SCORE SEARCHING

Solving C@merata queries involve challenges in both the interpretation of natural language concepts and in music analysis. Music XML score representation is poorly structured. The lack of basic one-to-one correspondences from score objects to Music XML entries is at the root of many problems. One other requirement is that score entities are modeled in an objective way, a requirement not met by the Common Music Notation part of MEI [6].

The following points should be considered in the design in the future of C@merata. They focus on acquiring experience on the fundamental fabric of music scores.

1. Voice leading: Explore the applicability of the rules of counterpoint. Initially use only vocal or other scores with one voice per stave. Focus the tasks on the first four species of counterpoint, by basing questions on cases such as parallel fifths and octaves, exposed octaves, sequences of thirds and fifths, false relations and others.
2. Harmony: Explore the problems of identifying chords with foreign notes present.
3. Devote only a small number of questions to keyboard scores.
4. For keyboard scores provide questions to identify the underlying harmonies in broken chords.
5. In keyboard scores use questions to explore underlying voice leading.

5. PRACTICAL CONSIDERATIONS

A number of steps are proposed on how C@merata might be improved in future.

1. Ensure Music XML sources are error-free: In both the 2015 and 2016 challenges many scores were found with errors and omissions. 14 of the 20 2016 scores could not be processed initially in CPNView. While it has not been possible to identify problems to date, it was possible to hand-edit some scores to make them useable. ‘Eroica’, ‘Weep, O mine eyes’ and ‘Am die Musik’ had no time signatures. An additional mid-bar measure number appeared in bar 5 of Chor002. In Sonata02-4 tuplets were not properly formatted, and sextuplets were incorrectly encoded; grace note heads appear as whole notes. Strange part and voice numbers appear in quartet10-3_m21i, Inven01_m21, quartet10-3_m21, quartet12-1_m21, sometimes in a form that looks like 32-character hexadecimal numbers, where numbers such as 1, 2 and 3 are normal in Music XML.

   The 2015 scores fared a bit better, but problems were found including scores with invalid bar lengths, missing rests, bar numbering and others.

   Much of this can be avoided by circulating scores to participants well in advance of the C@merata challenge so that feedback is available to correct and determine suitability. Previously used scores that have proved satisfactory could be re-used with new questions without compromising challenges in any way.

2. Gold Standard: An intensive study of the 2015 challenge questions revealed problems with the contents of the Gold Standard document: The answers for approximately 4% of the questions were incorrect, while an approximately 15% more valid answers were absent from the Gold Standard document. Careful visual inspection of each answer in the score in conjunction with adherence to the guideline document should minimize or eliminate the presence of incorrect answers if done in time, before the challenge is activated. The detection of missing answers will prove much more difficult. By publishing the Gold Standard following the deadline for submitting answers and giving participants a role in formulating revisions followed by re-scoring of all answers should be considered. Alternate to this is the evaluation of all answers that are classified as incorrect against the printed score.

6. REFERENCES

[1] Ó Maidin, D.S. 1995. A Programmer's Environment for Music Analysis. PhD Thesis, University College Cork, 1995.

[2] Ó Maidin, D.S. and Cahill, M. 2001. Score Processing for MIR. In *Proceedings of the International Society for Music Information Retrieval*, Bloomington, Indiana 2001, pp.59-64.

[3] Ó Maidin, D.S. 1998. A Geometrical Algorithm for Melodic Difference. In *Computing and Musicology II. Melodic Similarity. Concepts, Procedures, and Applications*, ed Walter B. Hewlett and Eleanor Selfridge-Field (Cambridge Massachusetts and London, The MIT Press, 1998), pp. 65-72.

[4] Forte, A. 1973. *The Structure of Atonal Music*. (New Haven and London: Yale University Press, 1973).

[5] Sholes, P.A. 1974. The Concise Oxford Dictionary of Music. (London, Oxford University Press).

[6] Music Encoding Initiative Guidelines, Version 3.0.0. on http://music-encoding.org/downloads/guide