Examiners' commentaries 2015–16

CO3346 Sound and music – Zone A

General remarks

Overall performance on this paper was very good: there were a few weak papers and many strong ones, with an overall average of a high second. Almost all candidates obtained a pass mark, with one particularly strong candidate obtaining almost full marks for the paper.

What follows is a brief discussion of the individual questions on this paper, with guidance on the answers expected by the examiners.

Comments on specific questions

Question 1 Computational models of music cognition

A less popular question, chosen by fewer than half of the candidates.

Part (a) expected a response explaining at least that the most elementary n-gram model of melodic pitch structure – a monogram model where n = 1 – simply tabulates the frequency of occurrence for each chromatic pitch encountered in a traversal of each melody in the training set. Explanations of a more complex structure were also acceptable.

For Part (b), a table showing the distribution of observed pitches, which are A: 3/6; B: 1/6; and D: 2/6, obtained full marks.

Part (c) asks about the purpose of using a monogram model in generative mode. It enables the generation of sequences of pitches that are statistically similar to the input.

Part (d), required a distribution of notes in the pitch sequence being sampled. Key elements examiners were looking for were: a distribution; generating a random number to select from the distribution; and generating a sequence of a particular length.

Parts (e) and (f) are based on material covered in some detail in the subject guide. The purpose of a probe-tone experiment is to compare the output of computational models of pitch sequences to see which is most compliant with the expectations of a human listener. Such experiments usually involve a probe-tone task where listeners are asked to listen to a melodic context and state how expected they find the final note (on a scale from 1 to 7, for example). By varying the final probe tone (say between an octave above and an octave below the final tone of the context), expectations can be gathered for a range of notes in the same melodic context. The procedure is repeated for a number of different melodic contexts, and the whole experiment is carried out with several different listeners whose responses are averaged for each pair of contexts and probe tones. It is then possible to produce expectation predictions from each of our models for each pair of contexts and probe tones and compare these to the average expectations of our listeners.

Part (g) expected both input and output aspects to be considered. The input to the algorithm is a selection of music symbolically represented as notes with pitches and durations. The output is the key of the piece of music.

For Part (h), examiners were not expecting pseudo code, but rather a description of how the algorithm attempts to compare the input note data to expected data for all minor and major keys, weighted by the duration of the notes, then selects the highest correlation.

Question 2 Interactive sound using Pure Data

A popular question, selected by almost all candidates.

Most candidates gave a correct answer to Part (a), that the elements are an Object, a Message Box and a Toggle, in order from left to right.

For Part (b), a correct answer would be:

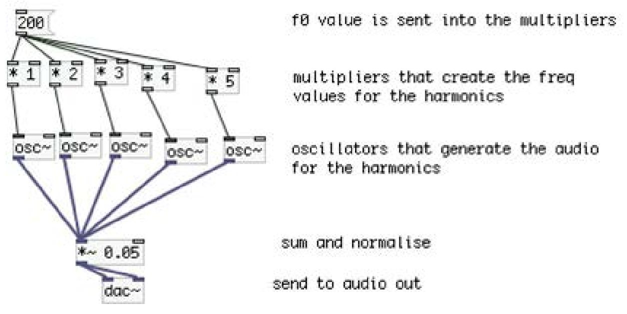
```
print:   6
print:   16
print:   36
print:   76
```

Not all candidates included the print, but the examiners awarded almost all the marks if the numbers themselves were correct.

Part (c) is discussed in the subject guide, and was straightforward.

The patch shown in Part (d) gives the sound of a sawtooth wave having its upper partials faded in and out in a kind of wah-wah sound. It uses subtractive synthesis, where a harmonically rich tone (the phasor) is passed through a filter; modulation is applied to the filter parameters.

For Part (e), any correct patch was acceptable, though the annotation was essential. The following is an example of a patch.


200
f0 value is sent into the multipliers
* 1
* 2
* 3
* 4
* 5
multipliers that create the freq values for the harmonics
osc~
oscillators that generate the audio for the harmonics
*~ 0.05
sum and normalise
dac~
send to audio out

Question 3 Algorithmic composition

This was a relatively popular question, with most answers being of good quality.

Parts (a) and (b) are based on the MIDI work described in the subject guide, and as such were straightforward. As an example, for part (a) (i), the MIDI channel's purpose is to address different instruments sitting on different channels; its range is 0–15 and it requires 1 byte to represent this. For part (b), it was essential to show your working, and explain the approach taken, as there are various ways of looking at the problem. One is to say that there are 2 messages with 4 bytes each, so 8 bytes is an obvious answer. However, another way to approach it is that there is a status byte containing the channel number and message type, as well as 2 data bytes for note number and note velocity. That makes 3 bytes per message, which is 6 in total.

Part (c) asked about the purpose of mapping when sonifying a particle system, which is to map the position of particles to parameters in the sonification engine.

Parts (d) and (e) were about mapping schemes for sonification. One example of a simple mapping technique, using only one characteristic, is to map the x-position of particles to a note number. This generates notes every time the values change in a quantized grid. Other examples were also acceptable. For the more complex scheme, examples could be density mapped to some timbral control, or velocity or acceleration, etc.

Question 4 Understanding musical interaction

This was also a popular question, chosen by many candidates.

Part (a) was generally well answered, though many candidates said that pitch was the same as frequency, not fully appreciating that though the two are connected, pitch is a perceptual thing. What follows are some good example answers that stress the important aspects of each element.

Rhythm concentrates on the time dimension of music, and is seen most clearly in drum and percussion music in general. It underlies nearly all Western tonal music, from Beethoven to the blues.

An individual note, say one note played on a piano, is associated with a given pitch. This is connected to the physical frequency of sound waves in the air. Each note on the piano or guitar has a different pitch, and the way that pitches are organised together is fundamental to many musics.

The same note or drum beat may be played louder or softer; this dimension of organisation is called intensity and is associated with the energy level of the sound waves involved.

For timbre, we are interested in the difference between notes of the same pitch and intensity played by different instruments, or sung by different people. The physics of this is more complicated; when guitarists change guitar between numbers, it is often about using a different timbre as part of a different atmosphere.

Surprisingly few candidates could accurately explain that a musical grammar is a formal definition of the rules and syntax for a type of music.

Part (c) is discussed in the subject guide. A grammar can be used for recognition and generation. Recognition means being able to state if a given piece of music has the correct syntax. Generation means that given a grammar, it is possible to generate automatically well-formed examples using that grammar. Simply saying recognition and generation, without explaining or discussing them, is not sufficient to obtain all the available marks.

For Part (d), examples could include the following, but other reasonable examples were accepted:

- Accompaniment: the auto chord player found on many types of synthesizer; or anything where there is a soloist and some (one/thing) else that plays the background tune
- Human/machine improvisation: swarm music with an audio input; live algorithms
- New interfaces to (new) instruments: extended instruments, such as sensors strapped into violins; live coding.

Part (e) required evidence of deeper understanding of how such a system might be implemented. Each point needed to be clearly stated and relevant. For example, in live coding, a technical challenge is to design a language that is suitable for editing running programs without re-compiling; an artistic challenge is to be able to program fast enough to make an interesting piece of music, etc.

Examiners' commentaries 2015–16

CO3346 Sound and music – Zone B

General remarks

Overall performance on this paper was very weak; many candidates failed to obtain enough points to pass, and the average mark across all candidates was a low third-class. This shows weak understanding of the material, and candidates are advised to ensure they cover all the material in the subject guide, as well as attempting all of the exercises and recommended reading.

The small number of candidates who performed very well, obtaining marks in the high first-class level (with one obtaining almost full marks) indicates that the standard of the examination itself was reasonable.

What follows is a brief discussion of the individual questions on this paper, with guidance on the answers expected by the examiners. Note that there is often not only one acceptable answer, and examiners accept any well-expressed response that addresses the question in an appropriate way.

Comments on specific questions

Question 1 Computational models of music cognition

This question was chosen by a good number of candidates; however, in general, the answers were weak.

For Part (a), a number of candidates confused digram models with diagrams, and gave irrelevant answers. In a digram model – where n = 2 – frequency counts are maintained for sequences of two pitch symbols, and predictions are governed by a first-order pitch distribution. This is derived from the frequency counts associated with only those digrams whose initial pitch symbol matches the final pitch symbol in the melodic context.

For Part (b), a table showing the distribution of observed pitches, which are A-A: 1/5; A-B:1/5; B-D: 1/5; D-D: 1/5; and D-A: 1/5, obtained full marks.

Part (c) asked about the purpose of using a digram model in generative mode. It enables the generation of sequences of pitches that are statistically similar to the input.

Part (d) required a description that includes selecting a note, and then selecting the next note by sampling the digrams starting with the selected note. It was necessary to be clear that A can lead to either A or B, etc., which not all answers included.

Parts (e) and (f) are based on material covered in some detail in the subject guide. The purpose of a probe-tone experiment is to compare the output of computational models of pitch sequences to see which is most compliant with the expectations of a human listener. Such experiments usually involve a probe-tone task, where listeners are asked to listen to a melodic context and state how expected, or fitting, they find the final note (e.g. on a scale from 1 to 7). By varying the final probe tone (say between an octave above and an octave below the final tone of the context), expectations can be gathered for a range of notes in the same melodic context. The procedure is repeated for a number of different melodic contexts, and the whole experiment is carried out with several different listeners whose responses are averaged for each pair of contexts and probe tones. It is then possible to produce expectation predictions from each of our models for each pair of contexts and probe tones and compare these to the average expectations of our listeners.

In response to Part (g), many candidates said that the Krumhansl–Schmuckler key-finding algorithm finds out which note appears most frequently. While this is part of how the algorithm works, it does not demonstrate sufficient understanding of what the algorithm actually is. Most candidates did know that it takes symbolic input.

For Part (i), an example input might be durations: (12d), where 12d is a 12-dimensional vector of durations of notes. It works (Part (j)) because the major and minor pitch profiles are derived from real examples of music.

Question 2 Interactive sound using Pure Data

This was a reasonably popular question, with slightly stronger responses from candidates.

Most candidates gave a correct answer to Part (a), that the elements are a message box, an element and a number box, in order from left to right.

For Part (b), a correct answer would be:

print : 8

print : 20

print : 44

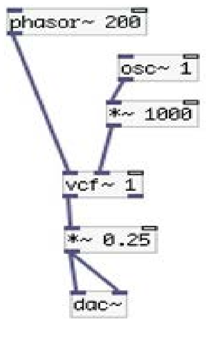
print : 92

Not all candidates included the print, but the examiners awarded almost all the marks if the numbers themselves were correct.

Part (c) is based on material from the subject guide, and was straightforward. However, many candidates simply drew a PD patch, even though the question asked for a description. Also, some candidates only considered one of the sample playback and the additive synthesis patches, instead of both as required. For sample playback, it is possible to use soundfiler or readsf to load a file. If the former, you would need to use a tabplay to play it back; if the latter, it is possible to play it back directly from the readsf. It is important to specify the file name, and the patch should feed the output to dac. For additive synthesis, there needs to be an addition of sinusoidal signals with different frequencies together, followed by normalising them and feeding them to the output.

The patch in part (d) uses FM synthesis. It would make a rich, thick tone as it has a fairly low frequency carrier wave. We know that it is FM synthesis as the frequency input of a carrier oscillator is being modulated by the amplified output of a modulator oscillator.

The following patch is one example of an acceptable answer, though annotation of the modulator, the oscillator, the filter and so on are required.



Question 3 Algorithmic composition

This was the least popular question, and those that chose it generally gave very weak answers, although one candidate gave reasonable answers.

All the material on MIDI, for Parts (a) and (b), is clearly presented in the subject guide, and the questions are straightforward ones. As an example, for Part (a) (iii), the MIDI controller number's purpose is to tell the instrument which parameter to change; its range is 0–127 and it requires 1 byte to represent this. For Part (b), it was essential to show your working, and explain the approach taken, as there are various ways of looking at the problem. One is to say that there is 1 message with 4 bytes, so 4 bytes is an obvious answer. However, another way to approach it is that there is a status byte containing the channel number and message type, as well as 2 data bytes for note number and note velocity. That makes 3 bytes in total.

Part (c) is again straightforward. The purpose of mapping when you are visualising a particle system, as in swarm music, is to map the position of particles to parameters in the visualisation engine.

For Parts (d) and (e), a simple mapping technique might use x and y values of particles to plot the particle system on screen, while a more advanced system might, for example, include using the velocity of the particles to draw a varying length tail.

Question 4 Understanding musical interaction

This question was not very popular though it was answered reasonably by most candidates who chose it.

Part (a) required specific examples of data rather than general descriptions. Some appropriate examples include the following:

- Rhythm: the BPM of a piece of music, or a metric pattern (e.g. 120bpm)
- Pitch: a note name, octave, duration and time offset (e.g. 64 for a MIDI note)
- Intensity: a note loudness value or measurements of loudness over time (e.g.127 for MIDI)
- Timbre: the name of the instrument, or the spectrum of the instrument (e.g. mfcc with 13 coefficients).

Part (b) required a justification, as well as the response of 'Yes'. It could be used to test if a given sequence matches that grammar. An answer of 'No' could also be justified, by the view that it is really just a single rule which is probably too simplistic for a full grammar. This gives a good illustration of why justification is essential for some responses.

Part (c) is straightforward. A grammar can be used for recognition and generation. Recognition means being able to state if a given piece of music has the correct syntax. Generation means that given a grammar, it is possible to generate automatically well-formed examples using that grammar.

Part (d) required an understanding of the process described, to come to the following conclusions:

- It does fit into accompaniment, as it is playing along with the instrumentalist.
- It does not really fit into human/machine improvisation, as the system is not really listening to anything the researcher playing; it only plays notes based on where she moves the violin.
- It fits the category of new interfaces to (new) instruments as it is a kind of 'hyper' or enhanced instrument interface.

For part (e), features to add to extend the violin system could include adding a pitch tracker so it played notes that complemented her notes; mapping the sensor to an effect or to timbre; or adding more sensors to generate polyphonic output or more complex timbre control.