Examiners' commentaries 2016–17
CO3346 Sound and Music – Zone A

General remarks

Overall performance on this paper was good: there were a few weak papers and many strong ones, with an overall average mark in the Lower Second range.

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

Comments on specific questions

Question 1 : Algorithmic composition

All candidates chose to answer this question, usually with good performance.

For part (a), examiners were expecting candidates to understand that while in normal composition, the composer directly selects the notes and other elements, in algorithmic composition, the composer defines a process through which notes, *etc*. will be selected.

Any three software systems for use in algorithmic composition were acceptable for part (b). Examples include Chuck, Super Collider, PureData, Max/MSO, CSound, Impromptu. Unfortunately, some candidates said that the Illiac Suite is a software system for use in algorithmic composition, which is not correct.

Not all candidates were able to give three clear reasons required for part (c). These could include any of the following: specialist environments provide more direct access to the tools needed for musical creation, such as sound synthesis and MIDI control; they provide built in data structures for representing musical information; and they can provide a musically-oriented workflow.

For part (d), a discussion of two properties of live algorithms was expected; however some candidates simply listed the four properties given in the subject guide, with no discussion. A strong answer would include two of the following:

The system must be able to reflect musically what it hears. The output must bear some resemblance to the playing of the human partner. At the simplest level, a direct echo would suffice, although this would become tedious.

The system must also be able to innovate. Its improvisations should suggest novelty; 'ideas' that the human partner can engage with. A quite random output would appear to satisfy this criterion, but would again become tedious because of the relative lack of predictability and patterning.

Autonomy, as opposed to automation implies that the system may play, or be quiet, at any moment: this is not contingent on the activity of the interacting partner. Automatic responses may lead to predictability and musical tedium. On the other hand, apparently random responses become unpredictable and uninteresting.

The system should have a degree of transparency. Its internal patterning should be apparent in the interpreted sonic output, and the relation between system response and musical input should not be too obscure.

Part (e) needed to relate to the two properties discussed in the previous part.

Some interesting and innovative responses were given for part (f). Improvisation involves an instrumentalist composing in real time on their instrument. Western classical music is more centralised, relying on a ready-made composition provided by a composer. Swarm music is closer to jazz improvisation because it is not composed beforehand, and utilises an algorithm which exhibits naturalistic, searching behaviour. One candidate observed that a common theme between jazz improvisation and swarm music is the lack of a conductor and the de-centralisation that relates to this in actual performance.

Question 2 : Musical Interaction

This was the least popular question, answered by less than a quarter of the candidates, with average performance.

For part (a), some candidates listed, but failed to describe, the characteristics as requested, thereby losing half of the marks available. As an example, for one of the characteristics, rhythm, a suitable description would be that it 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. Other characteristics include pitch, intensity and timbre.

For part (b), few candidates were able to explain clearly that metrical hierarchy is the hierarchy of intervals between events in the music. For example, the start of a bar could be one element, and the start of a beat could be another.

The metrical hierarchy, in part (c), should consist of several lines of dots, each line showing a different metrical division, up to the highest division required to match every note (bar, beat, half beat).

Some sensible answers were given for part (d); for example, find the smallest time interval between notes, count the number of notes happening at each integer multiple of that interval in the piece, select the division with the highest count of notes.

Correspondingly, a step by step version of the above, as a response for part (e), might be: set min to time interval between first two notes in the note list, iterate list updating min if between note interval is smaller. Multiply min by 1, count notes falling on those divisions, multiply min by 2, count notes falling at those divisions, *etc*. Finally, select the maximum of these, to give you the pulse.

Question 3 : Music Information Retrieval

This was a popular question, chosen by many candidates and generally answered well.

For part (a), as well as listing tasks, it was essential to explain why the task is useful. Candidates who only described the task did not get all of the marks available. Tasks and reasons could include:

- *music identification* — so you can identify a piece of music you do not know the name of, so you can buy it;
- *rights management and/or plagiarism detection* — so that record companies can protect their assets;
- *multiple version handling* — to group different versions of the same song together for analysis;
- *melody extraction and retrieval* — to allow computational musicology to be carried out on music;
- *performer or composer identification* — for historical music research;
- *style, mood, genre detection* — to allow creation of play lists of music that generate a certain mood in a club or cafe;
- *music-speech segmentation* — to make it easier to analyse mixed audio signals such as television.

Part (b) required a detailed description of the input, the approach to achieving the task, and the resulting output. In the example of music-speech segmentation, a comprehensive answer would include most of the following. Data input could be video of a television programme. This would be a compressed video file with an audio track. The audio track could be extracted from the video, and then features might be extracted, *e.g*. spectral features. Differentiation could then be achieved between speech and music based on these features. The output might be time data describing which segments of the video contain speech and which music. This is a content-based approach.

Part (c) was generally answered well, with candidates knowing that a cluster is a set of tracks with the same genre labels. The data that can be used to describe it might be the genre label plus a feature vector, *e.g*. distribution of MFCCs. The MFCC feature broadly describes the timbre of the music. To assign a piece of music to a cluster, the feature is extracted (*e.g.* MFCCs) and a distance metric is used to find the nearest cluster centroid. The genre of the tracks in that cluster is assigned to the track.

One problem is that the feature may not describe the genre well enough. For example, MFCC describes timbre. If the tracks in a genre have wide variation in timbre, mis-classification can occur.

Question 4 : Pure Data

This question was chosen by most candidates, but not always answered very well.

Part (a) required an understanding that Pure Data is a programming environment for computer music. It is graph-based rather than text-based (in contrast with most other programming languages or environments), and involves wiring boxes together to create programs.

A good answer to part (b) would demonstrate understanding that audio rate objects are running all the time, generating their output values at the sampling rate of the system. Control rate objects normally only generate an output in response to an input, and can only compute outputs at a maximum lower rate than the audio rate objects. For part (c), examples of audio rate objects include `osc~`, `dac~` and `vcf~` while control rate objects could be `metro`, `*` or `print`.

For part (d), labelling of the axes was required, which not all candidates included. The waveform should show two sine waves running at harmonic f0 and f1 of 220Hz.

Part (e) required both the description of the principle behind FM synthesis, and why it is computationally efficient. The basic principle is that one oscillator modulates the frequency of another oscillator, producing side bands in the frequency spectrum at its output. This results in a richer timbre than would be obtained by adding the two oscillators together. It is computationally efficient because it can be done using only two oscillators.

Finally, for part (f), the patch should include two `osc~` objects, one (the modulator) wired into the frequency input of the other (the carrier), after being scaled into audible frequency range and added to a fixed base frequency. The modulator should have its output multiplied into audible frequency range.