Neuro-Symbolic Composition of Music with Talking Points

Simon Colton,[1,2] **Berker Banar**[1] and **Sara Cardinale**[1]
[1]School of Electronic Engineering and Computer Science, Queen Mary University of London, UK
[2]SensiLab, Faculty of Information Technology, Monash University, Australia
s.colton@qmul.ac.uk

Abstract

We describe the Parley system, which generates musical compositions with the process foregrounded via talking points that help users engage. Parley employs a neuro-symbolic approach combining rule-based modules and pre-trained neural models to produce music in a communicable way, akin to standard composition approaches involving designing, writing, listening to, editing and analysing music. We highlight the potential of the system in a case study where users employ Parley's modules in a Colab notebook. To investigate the interplay of the rule-based and neural processes, we describe some experimental results comparing generated music before and after an editing process which employs a neural listening model.

Introduction and Motivation

There are many reasons why people compose music, only one of which is to have more music of high quality. We take inspiration from the YouTube broadcasts of well-known classical music composer David Bruce (youtube.com/@DBruce), where composition techniques are explained for educational and entertainment purposes, via the practice of composing new pieces of music, or explaining the processes leading to existing ones. In one episode, for example, he helps a novice composer to improve their first composition, offering numerous pieces of advice while composing an enhanced version, for the educational/entertainment benefit of the novice and the viewers on YouTube. In general, such advice on composition often falls into one of three categories:

• Following rules and heuristics of music theory and practice. Advice of this nature appeals to established rules based on agreed upon concepts, often phrased in terms of constraints, requirements and best practice.

• Making choices based on listening to the emerging composition. Choices don't have to be rule-driven, but rather composers are advised to try some alternatives and choose which sounds the best to them, in terms of the rest of the composition, or a particular genre or style, or general musicality.

• Striving for global properties of the composition. Strictly following rules can lead to tedious and repetitive music, so suggestions for introducing novelty and variety are often given. In contrast, composed music can sometimes lack coherence and structure, so advice on how to achieve consistency and regularity is also given.

We are building the *Parley* generative music system to compose music with a relatively transparent process which enables users to understand its process in a way that could potentially be entertaining or educational or both. We introduce the notion of a *talking point* as some piece of information that Parley can communicate in terms of rules employed, aesthetic preferences determined through listening or striving for a global property of the music it is generating. The purpose of a talking point is to communicate a decision made during a composition process or an opinion about an excerpt of music which provokes the user to engage with the music/process by possibly agreeing or disagreeing with Parley's opinion.

We have argued in (Colton and Banar 2023) that, while generative deep learning is the dominant force currently and undoubtedly produces highly impressive music (for instance see (Agostinelli et al. 2023)), it is not necessarily the best option if there are other reasons – such as education and entertainment – for the composing of music. This is, in part, because the black-box nature of deep learning makes it difficult to foreground decisions which have been made in the generative process. We suggest instead a neuro-symbolic (Hitzler et al. 2022) approach which employs rule-based and logical AI techniques in tandem with neural models to have both generative power and communicable decision making. With particular application to composition of music with talking points in mind, Parley employs rule-based approaches to **generate** initial compositions and neural *listening models* to estimate listener expectations and to tag excerpts in order to **edit** an evolving composition, as well as to **design** the overall form of the composition and **analyse** the music produced.

In future work, we will compare Parley with relevant prior work in computer assisted composition, including the PatchWork and OpenMusic projects (Assayag et al. 1999). We concentrate here on describing how Parley operates. In the next two sections, we describe the listening models employed, and the neuro-symbolic process in terms of a modular approach where users can employ multiple designer, generator, editor and analyser modules. We follow this with a case study, where a user co-creates compositions with Parley in the 'Flaneur' series, via multiple stages in a Colab notebook employing Parley's modules. We then describe some experiments which investigate the nature of the music produced with and without editing enabled by a listening model. We end by describing the general potential for neuro-symbolic approaches in computational creativity projects, and highlighting some future directions for the Parley project.

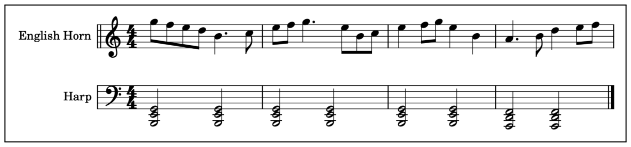


Figure 1: Example excerpt for the mood tagging experiment.

Neural Editing and Listening Models

Parley uses pre-trained neural models from two different projects, to perform functions comparable with human composers listening to their composition, as described below.

A Melody Pitch Class Prediction Model

It is standard to ascribe *the pitch class* of the note middle C – and any note which differs by this by a multiple of 12 semitones (an octave) – to be 0, with C# ascribed 1, D ascribed 2, and so on up the semitone scale. As described in (Banar and Colton 2022), an LSTM neural model (Hochreiter and Schmidhuber 1997) was trained to be able to predict the *interestingness* of each note's pitch class in a given piece of music. That is, the LSTM ascribes a score for each of 11 pitch classes, based on a window of notes preceding it. The more *unlikely* an observed pitch class is, according to the activations of the model in the output layer, the more interesting it is, and this process was used to help identify decision points in composed music. Two models were trained, on melodies written in the 18th and 20th Centuries respectively.

We repurposed the former of these models for Parley to use to edit the pitches in a melody to better fit the training set distribution of this pre-trained model better (i.e., with higher likelihood, lower interestingness). This could help produce melodies that adhere to traditional classical music expectations rather than more avant garde norms. In addition, as described below, choices Parley makes using this listening model can become the basis for talking points. Moreover, we describe below some experiments comparing the music generated before and after editing using this pre-trained model.

A Mood Tagging Model

Parley also employs the *Jamendo MoodTheme* neural model provided in the *Essentia* package (Correya et al. 2021), produced using TensorFlow (Abadi et al. 2015). Given an input audio file, this model has been trained on the MTG-Jamendo dataset (Bogdanov et al. 2019) to output a sequence of vectors, each containing 50 activations (as floats). Each activation refers to a different mood/theme tag and estimates the level of that mood expressed in the audio file. The mood tags include general emotional words such as 'happy', 'sad', 'dark' and 'dramatic'; some associated with musical style or genre, such as 'groovy', 'ballad' or 'christmas'; some associated with settings, such as 'summer' or 'space', and some associated with use-cases for the music, such as 'commercial' or 'trailer'. Sampling an audio file at 32khz, each sample is first passed through a pre-processing model, which encodes it into a latent space, with the resulting latent encoding input to the MoodTheme model. To produce an overall activation vector for an audio file resulting from a passage of music, the activations for each sample can be averaged.

threshold	0.5	1.0	1.5	2.0	2.5	3.0
average tags	8.5	4.8	2.7	1.5	0.8	0.5
zero tags %	4.4	15.9	35.7	54.8	70.5	81.3
ten+ tags %	32.1	12.7	4.5	1.8	0.6	0.2
most used %	28.9	16.0	8.7	5.4	3.1	1.9
tag	'dark'		'space'			'trailer'
least used %	19.5	10.0	5.4	3.0	1.2	0.5
tag	'deep'			'emotional'		

Table 1: For thresholds 0.5 to 3.0, average number of tags per excerpt; % excerpts with no activations for any tag over the threshold and more than ten tags over the threshold; highest and lowest % of excerpts a single tag is assigned to.

To leverage the power of the Jamendo MoodTheme model, we used an early version of Parley to generate 5,000 four-bar excerpts of music recorded in WAV files, then passed each through the model and recorded the average activation vectors produced. Each excerpt comprised a polyphonic melody part and a chord part with three notes which accompanies the melody, as per figure 1. We generated the specifications for Parley randomly, to produce diversity in the generated excerpts, by varying the following:

- The two instruments used for the melody line and chord accompaniment, with each ranging over all 110 non-percussion instruments in the FluidSynth soundfont (see below).
- The volumes (dynamics) of each instrument, ranging over midi volumes 60 to 127.
- The nature of the melody in terms of backbone and passing notes (see below), which changes the number of notes, rhythms and pitch ranges of the notes.
- The tempo of the excerpt, ranging from 1.5 seconds per bar to 3 seconds per bar.

The mean and standard deviations of the activations for the 50 mood tags over the 5,000 excerpts were recorded for use by Parley to produce talking points. In particular, we wanted it to be able to tell when a particular passage of music was exceptionally exhibiting a particular mood. To do this, we looked at the distribution of the model's activations over the 5,000 excerpts, and assigned a mood tag to an excerpt only if the activation it achieved for that tag was T times the standard deviation (for that tag) more than the mean (for the tag), for different thresholds T. The results for thresholds ranging from 0.5 to 3.0 are given in table 1. We see that, if the threshold is set to 0.5 standard deviations above the mean, then excerpts gain 8.5 tags on average, with this decreasing to 0.5 tags for a threshold of 3.0. Moreover, when a threshold of 2.0 is used, more than half the the excerpts are assigned no tag, and the most popular tag (namely 'space') is assigned to only 5.4% of the excerpts, hence no single tag is particularly over-used. Based on these findings, we have implemented in Parley the ability to tell whether a musical passage is *somewhat*, *quite* or *exceptionally* exhibiting a mood if the passage's activation for that mood is 0.5, 1.0 or 2.0 standard deviations above the mean respectively. This means that, referring to table 1, if a passage of music is similar to the 5,000 used here, at most 5.4% will be deemed to be exceptionally moody, 16.0% will be quite moody and 28.9% will be somewhat moody, which seems appropriate.

While talking points needn't be perfectly accurate, and users may choose to disagree with them, they must be suitably sensible. Hence we investigated the soundness of the tagging process over the kind of music that Parley can currently generate. To do this, we calculated correlations of activations over the 5,000 excerpts for pairs of tags and inspected the results. We found these pairs to be most negatively correlated, with correlation coefficients less than -0.5:

---

calm & epic; action & calm; action & relaxing; dark & romantic; epic & soft; calm & trailer; action & meditative

---

and we found these pairs to be the most positively correlated, with correlation coefficients above 0.9:

---

dramatic & trailer; epic & trailer; party & sexy; funny & upbeat; groovy & upbeat; melancholic & sad; energetic & fast; action & trailer; dramatic & epic

---

As can be observed, most of the pairs here are reasonable and expected. We did find some unusual results, such as 'sad' and 'happy' having a positive correlation of 0.50 and 'fast' and 'slow' with 0.35. However, in general we found the results sensible and reliable enough for use in Parley.

Neuro-Symbolic Composition

Parley is a modular system with which users can construct a workflow of modules which design, generate, edit and analyse musical compositions, output as both audio files (MP3 and WAV) and as PDF scores which could be played by musicians. Some modules can foreground decisions they make, and analysis modules examine the music and communicate talking points through both text and colour changes to the composition's score. The system is implemented in Python and can be embedded in Colab notebooks (Bisong 2019). This affords a programmatic and a limited GUI interface to Parley, and different notebooks can be used to produce different workflows and hence different compositions. Each notebook first downloads the Parley code repository before using the modules from it.

Cells in a notebook can expose parameters that guide various modules, and then can be used to run those modules, outputting talking point texts, images of the evolving composition's score, midi and audio files. Parley has an internal representation of notes, bars, chords, volume and tempo settings, etc., which can be translated into various formats, including Midi for use in third party music playing apps. It uses the `mido` Midi-handling python package (github.com/mido) and the Fluidsynth (fluidsynth.org) synthesizer to generate WAV files and `ffmpeg` (ffmpeg.org) to turn these into MP3 files. Parley can also output compositions represented as MusicXML (musicxml.com) files, and then employ the `music21` python package (web.mit.edu/music21) and a command-line version of MuseScore (musescore.org) to generate score PDFs. It further uses the `pdf2image` package (pypi.org/project/pdf2image) to turn PDF files into images to display in the Colab notebooks.

Music generated by Parley is episodic, to increase variety, and to enable changes at precise times to accompany other media, e.g,. as a soundtrack to a given video (Colton and Cardinale 2023). The designer, generator and editor modules produce (or change) the represented music for a given episode, with designers producing a skeletal form rather than playable music. Analyser modules produce text describing an episode, communicated in the notebook as talking points. Each module requires a user-supplied *specification* object describing how it should make choices in its processing. There are functions in Parley's codebase to copy and transform specification objects and to copy all specifications for an episode, so that longer compositions can be defined fairly easily.

Each specification object comprises a series of *parameters* which can be set to specify a module's processing. Users can introduce variety and control via the syntax used to represent a parameter. The most straightforward way is to just define the value of a parameter as an integer, float or string. However, parameters can also be defined probabilistically, with a set of values to choose from, each given with a percent probability. For instance, instead of specifying the integer 4 for a parameter, p, the user can specify instead: '3(pc=20) 4(pc=60) 5(pc=20)' for p, indicating that whenever it needs a value for p, it should choose one from $\{3, 4, 5\}$ with probabilities $\{0.2, 0.6, 0.2\}$ respectively. In conjunction with this, users can specify special exceptions, for example in the first (or last) bar of an episode or composition. For instance, p could be specified as '3(pc=20) 4(pc=60) 5(pc=20) 1(cb=1,cb=-1)' with cb standing for (c)omposition (b)ar and following the python standard of -1 indicating the last index of a list. This therefore specifies that p should be set to 1 for the first and last bar of the composition.

Users can also specify cycles for a parameter, for instance '3(ebc=1) 4(ebc=2) 5(ebc=3)' dictates that the value should be 3 in bar one, 4 in bar two and 5 in bar three of every three-(b)ar (c)ycle in the (e)pisode. Finally, users can specify that a parameter should range from one value to another smoothly during an episode. For instance '3(ebp=0.0) 4(ebp=0.8) 5(ebp=1.0)' specifies that the parameter should start when (e)pisode (b)ar (p)roportion is 0.0, rise to 4 when the episode is 0.8 complete, then to 5 when the episode ends.

In the following subsections, each of Parley's modules is detailed with (i) an overview of its purpose (ii) which parameters can be specified and how they affect the process, and (iii) how certain decisions (if any) are foregrounded.

Designers

Parley's designer modules enable users to plan the nature of their composition in advance of generating music for it. In particular, the designers afford description of the structure of episodes (form), the timing of the start and end of bars (tempo) the midi instruments to be used (orchestration) and the chord sequence which will underpin the music.

• The **Form Designer** module takes a string parameter specifying the number of different episodes a composition will have, and the order in which they will play. For instance, the value ABA specifies that the overall composition will comprise two episodes, A and B, with B sandwiched between two A episodes. The two A episodes will not contain exactly the same music, but will be generated by the same modules with the same specifications, thus subject to random variation. After bar timings for each of the episodes have been generated, the form designer constructs *episode objects* to contain bars, and a *form object* to contain the episodes.

• The **Bar Timing Designer** is parameterised with details of (a) the required overall duration (in milliseconds) of the composition (b) the number of episodes required, and (c) the desired duration (in ms) of the bars at the start and the end of each episode. Given these, it calculates the number of bars per episode, and the start and end timings of each bar. It also translates this into ticks for the midi generation (with 960 ticks per second). Details of the calculations for this are given in (Colton and Cardinale 2023). Users must specify the timings (in fractions of bar durations) that chords will change during a bar, using the rhythm format described below.

• The **NRT Chord Sequence Designer** can construct a sequence of trichords (i.e., with three distinct notes in) by employing operators inspired by the music analysis techniques which comprise Neo-Riemannian Theory (NRT) (Cohn 1998). The operators are themselves compounded from smaller units, and this module can be parameterised by the minimum and maximum length of the compounds. Generation of the chord sequences is done by randomly choosing an operator within certain constraints. In particular, the user can specify a *fixed key signature* constraint, so that an operator is only used if it outputs a trichord with all notes in a particular key. Another parameter specifies a *focal pitch* which produces an inversion of the chord such that all the note pitches are as close to this focal point as possible. This module generates a chord sequence to fit the chord change structure specified by the Bar Timing Designer. More details of NRT-based chord sequence generation are given in (Cardinale and Colton 2022) and (Colton and Cardinale 2023).

• The **Orchestration Designer** module is able to take a user-given specification of a mood, such as *happy* or *dramatic*, and produce the details of a pair of midi instruments which increases the likelihood of the generated music evoking that mood. To do this, it uses the specification of the instruments in the file containing 5,000 generative specifications described above. That is, for a given mood tag, T, and a user-specified range, R, this module finds the generative specifications which achieve the top R activations for T, chooses one randomly, and extracts the Fluidsynth soundfont midi instrument number for the melody and chord parts.

Generators

The generator modules in Parley are employed after instruments, episodes, bars and a chord sequence have been designed as above, and adds notes to the bars, sometimes on top of existing notes in a bar. The notes are represented internally with a multitude of properties including pitch, start tick, duration, volume and annotations for the written score, e.g., an indication that a note should be played staccato.

• The **Rhythm Note Sequence Generator** adds a sequence of notes to bars based on a specification which includes details of a rhythm. Such rhythms are described in terms of fractions of a beat for both the start of notes and their duration. They also specify which tone in the relevant chord underlying a bar should be used for the pitch. For instance, the parameter $1 : 1/4 : 1/4, 1 : 3/4 : 1/4$ specifies that in each bar, two notes should play, and each should be the first tone of the chord in the chord sequence at the moment the note plays. The first note starts on the first of four beats in the bar and has duration one quarter of the bar's duration. The second note starts on the third of four beats and also has duration one quarter that of the bar. Users can also specify whether notes should differ from the underlying chord tones by a multiple of 12 semitones (an octave). Normally, multiple note sequence generators are employed to produce chord accompaniments to a melody, as per the case study below.

• The **Voice Leading Melody Generator** adds notes to a bar in three stages. Firstly, guided by the pre-generated chord sequence, it adds chord tones to every bar as the *backbone* of the eventual melody in that bar. The user specifies how many backbone notes per bar, and the sequence is calculated so that no note repeats the one preceding it, if both are mirroring the same chord. When moving to a new chord, repetition is allowed, but the user can specify a *repetition policy* which can disallow/allow this and either (a) simply repeat the note (b) make both notes staccato to introduce a novel rhythm (c) tie the notes into one long one or (d) change the second note to a rest, again to vary the rhythm. This module creates voice leading melodies (i.e., which can be relatively easily sung (Aldwell and Schachter 2010)) by generating only intervals of a tone or a semitone. To minimise the number of pitches between two backbone notes, while adhering to the repetition policy, the backbone sequence is generated to minimise the average difference between the pitches of the notes. In the second stage, passing notes are added between the backbone notes, guided by a user-given *passing notes policy*. This can specify that all or no notes between two backbone notes are added, or only the one note with pitch closest to the midpoint between the backbone pitches is added. The user can specify that the passing notes must all be in a fixed key signature, but if they do not, then the sequence of passing notes changing by a semitone is used to bridge every pair of backbone notes.

In the third stage, the duration of each note is determined using a quantization process. Here, each note is initially given an equal duration of the fraction $\frac{1}{2^n}$ of the bar's duration for n as small as possible. This leaves a shortfall of duration to make up, and the module does this by randomly choosing a backbone note to extend the duration of by another $\frac{1}{2^n}$ until the total duration of the notes adds up to the duration of the bar. For instance, if the backbone and passing notes in a bar comprise 10 notes, then each note is originally given duration $\frac{1}{16}$th of the bar's duration, as this is the largest possible without running over the bar's duration. This produces total note duration of $\frac{10}{16}$ of the bar's duration, so a sequence of 6 backbone notes are randomly chosen (with multiplicity) and extended by $\frac{1}{16}$ to make up the shortfall. Recall that, to increase variety in the music, the specification of these policies can include multiple different choices chosen randomly and/or according to cycles or special cases for particular bars.

• The **Melody Harmonisation Generator** takes a given sequence of notes, usually a melody line, and to each note, n, adds another note with the same starting point in time and the same duration. The pitch of the added note differs to that of n by an interval which is chosen from a user-supplied set of options. The user can specify that the new note pitch must be in a particular key signature, and if so, when none of the intervals achieves this, whether to avoid adding a note or to choose the nearest in-key pitch available. The user can further specify a range of pitches that the new note must be within, and which types of notes to add new notes to, either *all* notes, *passing* notes or *backbone* notes.

Editors

Parley's editor modules take existing compositions and make alterations in the hope of offering improvements.

• The **Note Likelihood Editor** uses the LSTM neural model described above to make predictions about note pitch classes. When it is possible to improve a note's pitch class according to the model, i.e., to be more likely with respect to the training distribution of 18th Century melodies, this is undertaken. Running through all the notes, n, in a user-chosen melody line, the neural model takes as input a sequence of pitch classes from the notes preceding n, and produces a likelihood for the 11 pitch classes available for n. Working at the episode level, the user can specify how many preceding note pitch classes are in the sequence input to the model, with a default of 50 working well. With notes at the start of an episode that do not have 50 preceding notes, the notes at the end of the episode are wrapped around to provide these.

When a pitch class for note n is edited to a new one, the actual pitch must be chosen, and the user can specify how the module should do this. One method involves choosing the appropriate pitch closest to the existing one, while another method chooses the pitch closest to the pitch half way between the preceding and following note pitches (which produces smaller intervals). If a note is given a new pitch class which is more likely than it's existing one, it is marked in green on the score, or blue if the edited pitch class is the same. The user can parameterize this editor with a repetition policy similar to that for the Voice Leading Melody generator, which determines whether an edited note which repeats the pitch of the preceding one, is allowed or disallowed, and if allowed, whether it should be tied, repeated, turned into a rest or made staccato. In practice, when repetitions are not allowed, this means that some notes need to be edited to have a pitch with a lower likelihood than the existing one, in which case they are marked in red on the score.

• The **Orchestration Editor** can take an existing composition and change the midi instruments for individual parts, applied to specific episodes or the composition as a whole, as specified by the user. Alternatively, the user can choose one of the Jamendo MoodTheme tags, and, as per the Orchestration Designer, the module will find pairs of instruments which increase the likelihood that the composition reflects the tag's mood, with the instruments assigned to the parts in the composition, again specified by the user.

Analysers

The analyser modules in Parley serve the purpose of producing talking points that help the user to engage with the music as it is evolving and the processes which are being employed to design, generate and edit the music. Our conception of talking points is evolving along with the development of Parley, and currently includes (a) questions about whether the nature of a passage of music is good enough for the user, which may prompt them to change some aspect of the specifications and re-run a module (b) information about how the process of a module works (c) the identification of certain passages and foregrounding of them by text descriptions and on-score colour changes.

• The **Chord Repetition Analyser** can determine places in a chord sequence where the same chord is repeated, possibly too often. This is subject to a user given window (number of bars) in which to check for repetition, which moves from bar to bar. The module uses the entropy of the chord sequence in a window, and reports the windows with lowest entropy. It combines any overlapping windows into longer ones to report, and marks a user-specified number in red on the score.

• The **Discordancy Analyser** reports any combination of notes in the the composition which are playing at the same time and could be seen as being discordant (which the user may want to avoid for some compositions). The user specifies how long the overlapping notes should play for, in milliseconds, and what intervals should be marked as discordant. The interval of a semitone, i.e., an interval of 1 midi pitch between two notes is usually most noticeable, but an interval of 11 pitches can also be jarring. The user specifies a score for each interval and a total score that must be achieved before a discordancy is reported. The module highlights discordant moments by marking the notes in red.

• The **Likelihood Improvement Analyser** works on compositions, or episodes thereof, on which the LSTM Note Likelihood Editor has been used. It finds those bars where the cumulative improvement (according to the model) is the highest, subject to a user-given threshold of number of edited notes in the bar. Improvement is in terms of total ranking increase from the original pitch class to the edited one, as ranked by the likelihoods returned by the LSTM. This module highlights the chosen bars with purple notes, to contrast the red, green and blue added by the editor.

• The **Mood Highlight Analyser** employs the Jamendo MoodTheme model described above to process the audio from an episode and determine which mood tag is most applicable for it. If the activation of the tag is greater than 0.5 standard deviations (stds) above the mean (over the 5,000 excerpt database), it is reported. The module uses the words *somewhat*, *quite* and *exceptionally* for emphasis if the activation is greater than 0.5, 1.0 or 2.0 stds respectively. The module also finds the passages of b bars in an episode, where the episode's mood tag is most expressed, and reports the k most expressive, with b and k specified by the user.

Case Study: ‘Flaneur’ Compositions

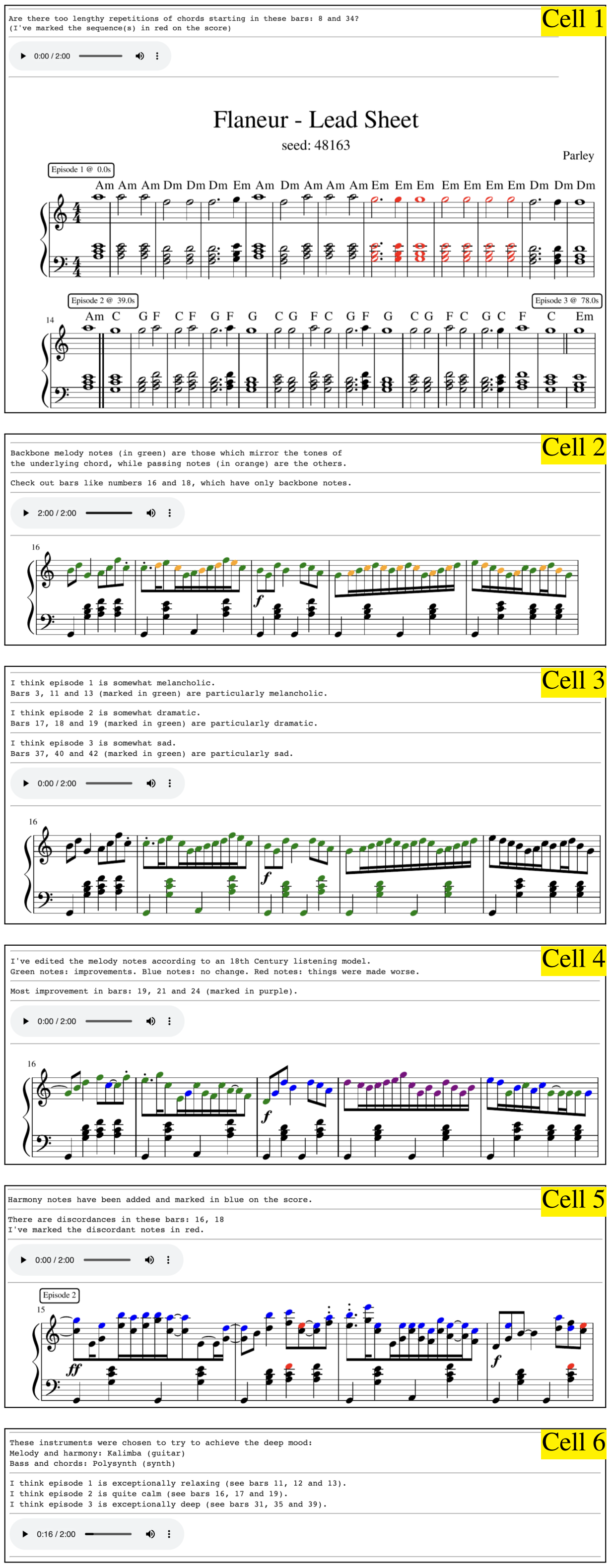
The Parley system currently exists as a codebase which can be downloaded from a repository into a Colab notebook. Each cell in the notebook can set parameters for one or more of Parley’s modules, which can then be run. In practice, a user exposes those parameters of the generative specifications which can fruitfully be experimented with in textbox/slider/checkbox GUI elements, and uses the modules to generate a piece of music in stages, showing audio and score representations of the evolving music at each stage. The same (or other) users can then later vary the parameters and use the notebook to produce novel music.

We used this approach to produce compositions in a series called *Flaneurs*, meaning “connoisseur of the street – a highly observant urban wanderer who takes in everything they see as they seek experiences that fuel their creative minds” (flaneurlife.com/flaneur-meaning). The idea was to juxtapose slower, more melancholy episodes, A, with faster, more upbeat episodes, B, with the music always ambling in nature, constantly in motion. The notebook is available at `https://tinyurl.com/3yjyjtrt`. Sample text and score outputs from the six cells of the notebook are given in fig. 2. The case study can be reproduced and experimented with, using seed 48163 in the Colab notebook.

In cell 1 of the notebook, a *lead sheet* is generated which comprises details of the design of the composition, namely the episode and bar structure and timings, and the chord sequence. The user can specify details for this, with exposed parameters including the overall episodic form, with default ABA given, as well as the composition title, duration in seconds and random seed, if the user wants to recreate a previous session. The parameters for the bar tempos at the start and end of episodes A and B are available to change. Likewise, the chord sequence generation specifications for A and B episodes are also exposed, with default settings for both episode types fixing chords to all be in the key of C major, but with A episodes requiring minor chords (hence fixing chords to Amin, Emin and Dmin) and B episodes requiring major chords (C, F and G), as per the overall idea for the compositions.

In figure 2, we see the cell output showing the episode start times and bars marked, and the chord sequence given. While only the design of the composition, it can still be played audibly, with a piano providing all the parts. Parley has provided one talking point by using the Chord Repetition Analyser to question whether the sequence of four bars of E minor chords is too long. This may prompt the user to change the specifications and re-generate the lead sheet. Note that episode 2 never has two repeating chords in a row, as the `max_repetitions` parameter was set to 1 for the NRT Chord Sequence Generator module of episode 2.

In cell 2, the user specifies the way in which the chords are to be played and the way in which the melody will be initially generated. For the chords, default specifications for usage of Parley’s Rhythm Note Sequence Generator are provided, for example the tonic of the chord is specified to be played with this rhythm: ‘1:3/8:2/8,7/8:2/8(cbc=0,cbc=1,cbc=2) 1:3/8:2/8,5/8:2/8,7/8:2/8(cbc=3) 1:3/8:2/8(cb=-1)’.



```
Are there too lengthy repetitions of chords starting in these bars: 8 and 34?
(I've marked the sequence(s) in red on the score)
```

```
Backbone melody notes (in green) are those which mirror the tones of
the underlying chord, while passing notes (in orange) are the others.

Check out bars like numbers 16 and 18, which have only backbone notes.
```

```
I think episode 1 is somewhat melancholic.
Bars 3, 11 and 13 (marked in green) are particularly melancholic.

I think episode 2 is somewhat dramatic.
Bars 17, 18 and 19 (marked in green) are particularly dramatic.

I think episode 3 is somewhat sad.
Bars 37, 40 and 42 (marked in green) are particularly sad.
```

```
I've edited the melody notes according to an 18th Century listening model.
Green notes: improvements. Blue notes: no change. Red notes: things were made worse.

Most improvement in bars: 19, 21 and 24 (marked in purple).
```

```
Harmony notes have been added and marked in blue on the score.

There are discordances in these bars: 16, 18
I've marked the discordant notes in red.
```

```
These instruments were chosen to try to achieve the deep mood:
Melody and harmony: Kalimba (guitar)
Bass and chords: Polysynth (synth)

I think episode 1 is exceptionally relaxing (see bars 11, 12 and 13).
I think episode 2 is quite calm (see bars 16, 17 and 19).
I think episode 3 is exceptionally deep (see bars 31, 35 and 39).
```

Figure 2: Text and score outputs from the six cells of the Flaneurs Colab notebook for composition with seed 48163. Bars 16, 17 and 18 can be compared as the complexity of the composition evolves.

This states that it should be played on the third and seventh of eight beats for bars 1, 2 and 3 of a 4-bar cycle, but on the third, fifth and seventh of eight beats for bar 4 of the cycle. A special case with (cb=-1) states that for the last bar of the composition, it should play only on the third beat. In every case, the duration is two eighths (one quarter) of the bar duration. Similar specifications for the other chord notes and a bass note are given as defaults in cell 2. The passing notes policy for A episodes is given by default as: ‘all(pc=60) mid(pc=30) none(pc=10)’, while for B episodes it is: ‘mid(pc=50) none(pc=50)’. This ensures a different feel to the two episodes, with larger intervals in B episodes. Parameters for volume changes over the two episode types is also exposed in this cell. The output from cell 2 is shown in figure 2, and we see that the score has green and orange notes highlighted to help the user understand the difference between backbone and passing notes, and Parley points the viewer to bars 16 and 18, where there are no passing notes, as talking points, or at least points of interest.

In cell 3, the user is able to choose instruments instead of the default piano, for all five parts currently in the composition. The generated music is analysed by the Mood Highlighter module and descriptions of the music in terms of mood are provided. As shown in figure 2, the user chose a pan-flute for the melody, keeping the piano accompaniment, and the mood for episodes 1, 2 and 3 were described by Parley as melancholic, dramatic and sad respectively. Note that instead of repeating the most activated tag for multiple episodes, it chooses the second or third as needed.

In cell 4, the Likelihood Improvement editor is used to change the melody line to be more expected in terms of the LSTM’s 18th Century music model. The user specified that all notes (backbone and passing) should be edited, and used the repetition policy: ‘tie(pc=60) staccato(pc=20) legato(pc=10) disallow(pc=10)’, to use when the edited note pitch is the same as the note preceding it. The notes which are changed to having a more likely pitch class (as per the model) are coloured green, while those which stay the same are coloured blue. For instance, we see in figure 2 that in bars 16 and 17, all but one note has been changed. In this cell the Mood Highlight Analyser module is further used to highlight in purple the three 1-bar passages where likelihood improvement is highest, with one such given in fig. 2.

In cell 5, the Melody Harmonisation Generator module is parameterised and employed to add a harmony line to the melody. The default pitch range for the harmony line is given as 60 to 90, so the pitch stays above middle C, the intervals allowed are largely above the melody line (4, 7, 9) with one below (-3), it is told that notes must be in C major, but not to map notes to a key signature, which means that some melody notes are not harmonised. The oboe instrument is used, as its reedy sound compliments the pan-flute. The harmonised notes are highlighted in blue on the score, and the Discordancy Analyser is used to highlight two points where notes differing by 11 semitones are played, as highlighted in figure 2. In the final cell, the user can once again change the instruments used and the volumes for each instrument, in order to produce a final composition. Alternatively, they can choose one of the 50 tags in the Jamendo MoodTheme model, and employ Parley’s Orchestration Designer to choose a pair of instruments, one for the melody/harmony and one for the bass/chord. In the case study, the user chose the tag ‘deep’, and the module chose the instruments Kalimba and Polysynth. The resulting episodes are tagged with talking points ‘exceptionally relaxing’, ‘quite calm’ and ‘exceptionally deep’, with relevant passages highlighted in each episode.

Overall, careful choice of the parameterisations of the generative specifications to produce Flaneur pieces leads to music which is, subjectively, *consistent* due to the repetitive nature of the bass and chords, *varied* (due to the episodic nature, passing note and repetition policy), surprising (due to the likelihood editing), *rich* (due to the harmonisation) and *evocative* (due to the mood-based orchestration). Two 10-minute Flaneur compositions were performed in public in February 2023, as part of a public engagement event at Queen Mary University of London.

Experiments with the Note Likelihood Editor

To investigate the way the likelihood editor changes melodies, we used 100 random seeds to each produce 16 Flaneur compositions via variations on the default generative specifications, and without the harmonisation or instrumentation changes. The melody generation was varied with the passing note policy ranging over ‘all’, ‘mid’, ‘none’ and ‘all(pc=33) mid(pc=33) none(pc=34)’ which we called ‘mixed’. For each policy, we produced four pieces: (i) a voice-leading (VL) version without editing (ii) an edited version (L+) where the process is also constrained to maintain the backbone quality of backbone notes (iii) an edited version (L-) without this constraint, and (iv) a version where a note’s pitch class is chosen (R)andomly, but must be in C major. For the L+ and L- versions, pitch classes were also constrained to C major.

We then measured numerous qualities of the generated melodies, in particular certain entropies, with the results given in table 2. The first measurement was the average improvement in the ranking of edited pitch classes over the original, as per the model’s calculated likelihood. We see that, for all the L+ and L- versions, the ranking improves by around 5.5, which is half the maximum it could be, of 11. Hence, even with the backbone and C major constraints, the editing is still effective. We next see that the entropy over the pitch classes is high (above 0.9) for all the setups, hence predicting a next note’s pitch class is difficult. The same is true for the interval between notes, with the exception of the (unedited) voice-leading compositions with passing notes, where the entropy is around 0.6 to 0.7. Subjectively, these pieces are somewhat dull, as the voice-leading melodies are rather predictable. When edited, the average interval entropies in these compositions rises to 0.9 and above, and – again subjectively – are more varied and surprising. However, the interval entropy for edited melodies is consistently lower than that for the randomly edited melodies, and we have subjectively noted that, while more surprising than pure voice leading, the edited melodies do not seem random.

We also measured the average and maximum *run length* in a composition’s melody, with a *run* defined as a sequence

	All Passing Notes				No Passing Notes				Mid Passing Notes				Mixed Passing Notes			
Measurement	VL	L+	L-	R	VL	L+	L-	R	VL	L+	L-	R	VL	L+	L-	R
Edit Improvement Av.	-	5.861	5.842	-	-	5.356	5.337	-	-	5.658	5.644	-	-	5.667	5.644	-
Pitch Class Ent.	0.938	0.939	0.935	0.996	0.962	0.918	0.917	0.990	0.952	0.933	0.934	0.995	0.954	0.935	0.930	0.994
Interval Av.	1.710	2.584	2.367	2.941	3.059	2.388	2.635	2.933	2.062	2.582	2.412	2.939	2.142	2.566	2.426	2.956
Interval Ent.	0.610	0.915	0.895	0.939	0.930	0.856	0.897	0.936	0.650	0.916	0.900	0.939	0.722	0.915	0.898	0.937
Run Length Max	7.000	10.660	23.450	5.980	4.770	7.820	10.970	5.230	6.930	8.490	18.610	5.840	6.900	9.270	19.790	5.650
Run Length Av.	4.078	2.535	2.616	2.481	2.525	2.476	2.524	2.470	3.551	2.512	2.597	2.486	3.413	2.513	2.582	2.472
Run Length Ent.	0.944	0.537	0.501	0.583	0.718	0.536	0.560	0.632	0.835	0.546	0.530	0.599	0.858	0.544	0.515	0.600
Voice Lead Av.	1.000	0.497	0.529	0.419	0.329	0.460	0.421	0.409	0.835	0.505	0.507	0.418	0.788	0.477	0.505	0.413
Voice Lead Ent.	0.000	0.994	0.979	0.978	0.914	0.987	0.955	0.973	0.644	0.986	0.973	0.978	0.736	0.988	0.977	0.977
Backbone Match Av.	1.000	1.000	0.450	0.425	1.000	1.000	0.499	0.424	1.000	1.000	0.452	0.430	1.000	1.000	0.463	0.426
Backbone Match Ent.	0.000	0.000	0.984	0.980	0.000	0.000	0.994	0.981	0.000	0.000	0.986	0.984	0.000	0.000	0.991	0.980

Table 2: Analysis of the properties of melodies averaged over 100 Flaneur compositions, for sixteen different generative setups.

of notes with only positive intervals (i.e., repeatedly going up in pitch), only negative intervals or entirely zero intervals (i.e., repeated notes). To measure the run length entropy, we recorded the lengths of each successive run in a composition's melody and calculated the entropy of this sequence. The results in table 2 help to quantify a phenomenon we observed in the Flaneur melodies after editing, namely long sequences of repeated notes. Recall that the LSTM in the Likelihood Editor listening model requires a seed melody to predict the likelihood of the subsequent pitch classes. As the window only moves by one note, it doesn't change greatly from note to note, hence likelihood calculations can sometimes differ very little over a series of notes, resulting in repeated notes.

The average maximum run length over the 100 L- melodies with all passing notes included was 23.450, which is inflated due to the repetitions. However, the repetitions are broken up with the requirement that backbone notes retain their backbone quality, as the maximum run length reduces to 10.660. Interestingly, while the maximum run length is higher for edited melodies than voice leading ones, the average run length is lower. This again indicates that melodies are less predictable, with many changes in direction and long passages of repeated notes. We also see that the more passing notes in a melody, the more notes in general, hence the higher the potential for repeated notes and increased maximum run length. As run length for edited melodies grows, entropy decreases, which can again be explained by the repetition of notes, rather than increasing or decreasing pitch runs.

In addition to clarifying the editing process, table 2 shows the wide range of entropies for melody properties achievable with certain choices in the generative setup. Hence, there is some evidence that users can control various entropies of the melodies produced, hence can vary melodic expectation (Pearce and Wiggins 2006), and we plan to explore this further. For completeness, we also measured the *voice leading quality* of each composition's melody, giving a score of 1 if a note's pitch is within one whole tone of the previous note's pitch, 0 otherwise. We also measured a *backbone match* calculation, scoring 1 for each backbone note which remained so after editing, 0 otherwise. The results from these evaluations largely conformed to our expectations, as per table 2.

Conclusions and Future Work

We have presented the Parley neuro-symbolic system as a series of generative and analytical modules able to compose music in stages while exposing decisions and opinions as talking points. As with (Aggarwal and Parikh 2020), we believe neuro-symbolic approaches hold much promise for more understandable and interesting generative AI systems. We plan to implement dozens more rule-based modules covering music theory and pre-trained neural models for analysing compositions. As above, the modules will be analogous to composers following rules/heuristics, listening to their evolving compositions and trying to balance global features of the music. We plan to integrate automated reasoning (Robinson and Voronkov 2001) and constraint solving (Rossi, van Beek, and Walsh 2006) techniques to help in the latter tasks.

We also plan to more formally define the notion of a *talking point*. We believe this is a valuable form of framing (Cook et al. 2019; Charnley, Pease, and Colton 2012) creative processes and outputs, which could improve the interactions that users have with a generative AI system, either in terms of casual creation (Compton and Mateas 2015) and entertainment, reflective creation (Kreminski and Mateas 2021) or education (Llano et al. 2020). Once Parley is more sophisticated (including a more intuitive user interface), we plan experiments with amateur and professional musicians and composers to get feedback on the talking points and on Parley in general.

We have already started handing over some creative responsibilities to Parley, for instance with the Orchestration Designer and Editor modules able to choose instruments to (try to) achieve a mood or theme. We are also experimenting with a module which can expand excerpts into entire compositions, following *mood arcs* where the strength of the mood expressed waxes and wanes over the episodic structure of the composition. We hope to investigate the value of neuro-symbolic music composition, where users follow fully (computationally) autonomous processes, with intrigue and explanation provided by talking points about Parley's own compositions, rather than those being made by a user. Ultimately, we aim for Parley to add to musical culture itself, as described in (Colton and Banar 2023).

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments, suggestions and pointers to relevant prior work. We would also like to thank the developers of the various software packages that Parley is built on top of. Berker Banar and Sara Cardinale are supported by the UKRI/EPSRC Centre for Doctoral Training in AI and Music (grant number EP/S022694/1) and Queen Mary University of London.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; and Zheng, X. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Aggarwal, G., and Parikh, D. 2020. Neuro-symbolic generative art: A preliminary study. In *Proceedings of the 11th International Conference on Computational Creativity*.

Agostinelli, A.; Denk, T. I.; Borsos, Z.; Engel, J.; Verzetti, M.; Caillon, A.; Huang, Q.; Jansen, A.; Roberts, A.; Tagliasacchi, M.; Sharifi, M.; Zeghidour, N.; and Frank, C. 2023. MusicLM: Generating music from text. *arXiv 2301.11325*.

Aldwell, E., and Schachter, C. 2010. *Harmony and Voice Leading*. Thomson Schirmer.

Assayag, G.; Rueda, C.; Laurson, M.; Agon, C.; and Delerue, O. 1999. Computer-assisted composition at IRCAM: From PatchWork to OpenMusic. *Computer Music Journal* 23(3):59–72.

Banar, B., and Colton, S. 2022. Identifying critical decision points in musical compositions using machine learning. In *Proceedings of the 24th IEEE International Workshop on Multimedia Signal Processing*.

Bisong, E. 2019. Google colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Apress.

Bogdanov, D.; Won, M.; Tovstogan, P.; Porter, A.; and Serra, X. 2019. The MTG-Jamendo dataset for automatic music tagging. In *Proceedings of the ML4MD Machine Learning for Music Discovery Workshop at ICML*.

Cardinale, S., and Colton, S. 2022. Neo-Riemannian Theory for generative film and videogame music. In *Proceedings of the 13th International Conference on Computational Creativity*.

Charnley, J.; Pease, A.; and Colton, S. 2012. On the notion of framing in computational creativity. In *Proceedings of the 3rd International Conference on Computational Creativity*.

Cohn, R. 1998. Introduction to Neo-Riemannian Theory: a survey and a historical perspective. *Journal of Music Theory* 42:167–180.

Colton, S., and Banar, B. 2023. Automatically adding to artistic cultures. In *Proceedings of the International Conference on Computational Intelligence in Music, Sound, Art and Design*.

Colton, S., and Cardinale, S. 2023. Extending generative Neo-Riemannian Theory for event-based soundtrack production. In *Proceedings of the International Conference on Computational Intelligence in Music, Sound, Art and Design*.

Compton, K., and Mateas, M. 2015. Casual creators. In *Proceedings of the 6th International Conference on Computational Creativity*.

Cook, M.; Colton, S.; Pease, A.; and Llano, T. 2019. Framing in computational creativity: A survey and taxonomy. In *Proceedings of the tenth International Conference on Computational Creativity*.

Correya, A.; Alonso-Jiménez, P.; Marcos-Fernàndez, J.; Serra, X.; and Bogdanov, D. 2021. Essentia TensorFlow models for audio and music processing on the web. In *Proceedings of the Web Audio Conference*.

Hitzler, P.; Eberhart, A.; Ebrahimi, M.; Sarker, M. K.; and Zhou, L. 2022. Neuro-symbolic approaches in artificial intelligence. *National Science Review* 9(6).

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Kreminski, M., and Mateas, M. 2021. Reflective creators. In *Proceedings of the 12th International Conference on Computational Creativity*.

Llano, M.; d'Inverno, M.; Yee-King, M.; McCormack, J.; Ilsar, A.; Pease, A.; and Colton, S. 2020. Explainable computational creativity. In *Proceedings of the 11th International Conference on Computational Creativity*.

Pearce, M., and Wiggins, G. 2006. Expectation in melody: The influence of context and learning. *Music Perception* 23:377–405.

Robinson, A., and Voronkov, A., eds. 2001. *Handbook of Automated Reasoning*. MIT Press.

Rossi, F.; van Beek, P.; and Walsh, T., eds. 2006. *Handbook of Constraint Programming*. Elsevier.