Jack 1: Pattern and Shape in Music Generation

Eliot Handelman and **Andie Sigler**
School of Computer Science,
McGill University
Montreal, Canada

Abstract

Jack 1 is an original meta-music piece which randomly builds a multilevel data structure to generate, extend and sequence musical shapes. Jack 1 is presented as an experiment in generating high-level, engaging music using simple ideas about pattern, shape and supershape, growth, and formal zoning.

Motivation

Jack 1 a musical meta-artwork which doubles as an experiment in using a (specific) small set of ideas for generating music, while excluding other ideas.[1] Jack 1 grew out of a analytical theory, which raised the question: if we know what kinds of things a given computational analyzer will look for, then why not try generating the right sort of output by using the analytical system backwards?

Jack 1 is an algorithmic system that produces music with an engaging animated quality, with motives, developments, formal thrust, sections and climaxes. Pieces of 90 – 300 seconds can hold together, presenting a coherent whole.[2] The pieces are not entirely "finished," coming out with flat velocity, but this is addressed in later work on automatic orchestration through automatic analysis.[3] Sample output of Jack 1 can be found at http://www.computingmusic.com.

In evaluating any system, it is a challenge to understand the interaction of different components. In a system with fewer (and less complex) components, it is easier to explore their individual contributions. The strength of a simple system may predict that well-performing, more complex systems could be simplified and therefore better understood. The shortcomings of a simple system may demonstrate the efficacy of excluded concepts, directing the addition of new modules.

In Jack 1, music data is completely excluded: no statistics or recombination of snippets from existing music are used. Jack 1 uses no explicit rules nor constraints about musical situations that must or may not occur. It has no analysis facilities at all, and therefore cannot recognize any musical situations. It does not perceive, remember, or learn. It uses no templates to delineate forms, phrases or other musical schemata. It has no theories or data about harmony, chords or chord progressions, keys, cadences, or voiceleading, nor about information theory or human expectations.

Part of what we can learn from Jack 1 is the degree to which many of the above facilities, though they may be desirable in a more complex musical experiment or meta-piece, may be unnecessary even for a high level of music-making – and that some of them may be generated through process, without explicit modelling. The only traditional music theoretic concept used is the diatonic major scale: not treated functionally with tonic, leading tone, and so on, but simply as a set of pitches (which, nonetheless, do have a structure and a history).

At a basic level, the functioning of any computational system can be divided into decisions that are made deterministically (given inputs, variable settings, and program state) and those that are made randomly (i.e. using a pseudorandom number generator). Decisions that are made randomly are, in fact, partially constrained, as when a random integer is called for within a given range. The role of randomness in a music generation system is essential for experimentation. Leaving some decisions open to randomness allows us to check the effect of other, deterministic decisions under a variety of conditions.

By aiming for a high degree of randomness, we can try to find a *minimum* set of conditions under which a score functions as music. This is parallel to Harold Cohen's question, "What is the minimum set of conditions under which a set of marks functions as an image?"[4] We note that Cohen uses the word *image*, perhaps as opposed to *work of art* – since art, and musical art, are radically open. But "image," for Cohen, is related to *human perception* (in particular, of *objects*, which may nonetheless be abstract in their identity). The artistic question posed by Jack 1's author is parallel in that the interest is not in making the most minimal possible music, but in exploring minimal parameters and concepts under

[1]Jack 1 was written by Eliot Handelman in 2009 and revived by both authors in 2013; this paper is the first public explanation of its functioning.

[2]There is no technological barrier to longer pieces; short pieces are an arbitrary choice for ease of listening.

[3]Handelman et al. "Automatic Orchestration for Automatic Composition." MUME at AIIDE, 2012.

[4]In "The further exploits of Aaron, painter." Stanford Humanities Review 4(2), 1995.

which a kind of music emerges that evokes specifically *musical* responses – in the narrower sense of music, as opposed to expanded musical or sound art.

Jack 1 requires no run-time human intervention. Therefore, an analysis of patterning functions with respect to their musical consequences is possible. In this paper, we describe a small set of concepts and their realization in a musical meta-piece.

Cyclers: extensible shapes

The fundamental mechanism of Jack 1 is the *cycler*. In effect, cyclers are machines for randomly generating patterns with different kinds of interacting regularities.

In particular there are four ways that cyclers are used in the meta-piece: cyclers that generate small musical *shapes* (one might say "motives," but this has historical connotations that we do not wish to defend), cyclers that cause small shapes to exibit *grow* behavior as they recur throughout a piece (as a stand-in for other concepts of variation or development), cyclers that create *zoned* dispositions of these small, growing musical shapes (as a means of formal organization), and cycler-patterned *supershape*, which organizes the pitch relations of the (growing, zoned) small shapes.

Here we describe three increasingly complex types of cyclers: the recycler, the aN-cycler, and the X-cycler.

The *recycler* is the simplest cycler.[5] It consists of a sequence of length r of terms, and a pointer keeping track of which term is currently addressed. *Terms* are symbols that can be used to refer to any object – in Jack 1, they can represent musical intervals, shapes (i.e. segments of music), or even other cyclers. Abstractly, we can represent these terms by letters of the alphabet. In a run of Jack 1, the term sequences for all recyclers is randomly generated (from a limited superset of options).

A recycler can be *queried* by sending it an integer i; it responds by emitting a sequence of i terms. The recycler simply starts from the term addressed by the pointer, and steps along its sequence emitting each term in order until i terms have been emmitted. In "loop" mode, the pointer loops back to the beginning of its sequence when it hits the end; in "sweep" mode it travels back and forth through the sequence. There are a few more simple commands to influence the recycler: a "reset" button which moves the pointer back to the beginning of the sequence; and a "reverse" mode, reversing the sequence.

More complex cyclers are built out of simple recyclers. The *aN-cycler* has a base recycler, and a second recycler where the terms act as counters for how many times to repeat each term in of the base recycler. "aN" refers to each term a of a sequence being repeated some number N times.

For example, if we had a base sequence $[a; b; c; d]$ and a counter sequence [4;3;2;1], then the default emitted sequence would begin $[a; a; a; a; b; b; b; c; c; d...]$. When the base sequence and the counter sequence have different cardinalities, this can result in the emitted sequence not repeating for a longer time. For example, the base sequence $[a; b; c; d]$ and the counter sequence [3;2;1] gives $[a; a; a; b; b; c; d; d; d; a; a; b; c; c; c; d; d; a...]$.

The *X-cycler* is significantly more complex, and hasn't been used to its full potential in Jack 1, but we explain it here to give a sense of what could be possible with cyclers. It remains to be explored to what complexity cyclers can be musically useful – this could be developed into a systematic psychological program exploring the limits of perceptible pattern complexity.

An X-cycler is composed of an aN-cycler and a group of recyclers. Recall that the aN-cycler consists of a base sequence and a counter sequence. Here, the terms of the base sequence refer to recyclers in the recycler group. The X-cycler runs by using the aN-cycler to address the recyclers: the base sequence chooses which recycler to address and the counter decides how many terms should be emitted from that recycler.

For example, suppose we are given an X-cycler: its aN-cycler has base terms [0;1;1] and counter [3;2], and its recycler group consists of recycler 0 with terms [a;b;c] and recycler 1 with terms [x;y;z]. Then if the X-cycler is queried to emit fifteen elements, the aN-cycler tells it to emit three elements from recycler 0, two from recycler 1, three from recycler 1, two from recycler 0, three from recycler 1 and two from recycler 1. Supposing the recyclers are set to start over from the beginning of their sequences when addressed, we get [a;b;c;x;y;x;y;z;a;b;x;y;z;x;y].

As we have seen, cyclers can control other cyclers, since the output of one cycler can be the input to another. The Jack 1 meta-piece consists of a structure of interacting cyclers. At runtime, each cycler in the structure is randomly instantiated. This means that its term sequence is randomly generated, some settings (e.g. loop mode or sweep mode) may be randomly set, and, in some cases, there is a randomized decision of which kind of cycler to use. Subsequent sections of this paper describe the meta-piece, and how it uses cyclers to produce *grow* behavior, *zoned* form, and *supershape*.

Once instantiated, each cycler has deterministic behavior. Therefore, all that all that is needed to (deterministically) generate a musical piece is to query the top-level cycler with an integer (which indirectly determines the length of the piece, since it determines how many musical shapes are ultimately generated). It's possible to save the cycler structure of a musical piece in order to determine precisely how it was generated; as well (although we haven't done this with any of the random Jack 1 pieces presented online), it would be possible to edit a structure of cyclers (e.g. by hand, or by randomly perturbing specific parts of it) to obtain variations of a randomly generated piece.

Shape and Super-Shape

In this section we discuss the lowest level cyclers, which generate local material, melodic "shapes" and chords. The atom here is the (diatonic) interval: the terms of the cyclers at this level represent intervals or chords (sets of intervals).

Jack 1 constructs random chords by randomly choosing how many intervals to use (e.g. between 0 and 5) and then randomly choosing intervals within (for example) one or two octaves of an initial pitch.

[5]A similar idea occurs in Rick Taube's CM program.

Jack 1 also constructs interval term sequences (i.e. melodic fragments) randomly; the cycler's generating process may constrain intervals to the set [-2;-1;0;1;2], or allow larger leaps. The sequences are generally four to eight terms long.

Eight to ten random cyclers emitting basic melodic shapes and chords are generated, and will be queried by higher level cyclers to form this basic material into a piece of music.

Jumping ahead to the end of the generation process – suppose we have generated a fixed sequence of small shapes, where each shape is a sequence of notes and/or chords expressed as *intervals*. The final step in the generation process is to assign each shape to a *pitch* level, in relation to the other shapes around it.

To do this, we use another cycler with interval terms to generate a *supershape*. After we choose an arbitrary pitch for the first shape, we use the intervals emitted by the supershape cycler to determine the relative pitch levels for the rest of the shapes in the sequence. In particular, we could take the *top* note in each shape, and assign these to the intervals in the supershape. We could also take the bottom note, the first note, or the last note, or some pattern of these options – where the pattern, of course, is emitted by another cycler whose terms represent the options *top, bottom, first, last*.

Growth

Querying a cycler (e.g. of intervals) in different ways can result in different, related shapes. In place of a theory of variation or development, we use cyclers to create sequences of related shapes that *grow*.

Suppose we have a cycler that emits a sequence of intervals. Now all we have to do is query it to emit 3, then 4, then 5, then 6, then 7, and so on, and we will have a *growing* shape. Most often, this is done with interval cyclers set to restart at each query, since this makes the relation most evident. It's easy to see how to make a shrinking shape, or a shape that shrinks for a while and then jumps back up in size and starts growing.

The "grow" cyclers that query the interval cyclers don't have a completely randomized sequence of terms, since we want growth to be at least somewhat linear. The terms of the grow cyclers are generated by taking an ordered series of intervals (i.e. [3;4;5;6;7;8;9]) and randomly selecting one or more varying functions on this, including random rotation (e.g. [7;8;9;3;4;5;6]) and partial reversal (e.g. [5;4;3;6;7;8;9]).

Jack 1 also uses the growth effect to produce flexible rhythm. Often, it will set each cycler query to take up the same amount of time, so that 3, then 4, then 5 events are fit into the same duration, giving an accelerando.

There are also more complex grow cyclers that have query access to more than one basic cycler, creating flexible compound shapes. In figure 1, we see the output of a grow cycler querying three cyclers, always in the same order. The basic cyclers respectively produce a chain of rising chords, a chain of decending intervals, and ascending chains of repeated notes. At each iteration, each one of the basic shapes

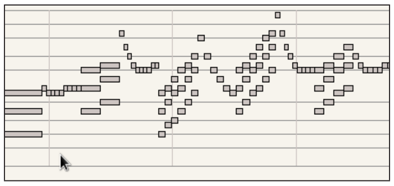

Figure 1: A compound shape with independently growing subshapes.

is (randomly) either a little longer, a little shorter, or the same length as in the previous version of the shape, while the duration of the whole remains constant.

Zones

Now that we have created a set of grow cyclers that (by querying interval and chord cyclers) emit growing shapes, it's time to use another cycler to sequence these growing shapes into a final piece. Here we introduce a new kind of cycler called the *zone cycler*, which is designed to produce flexible larger scale forms which have continuity and recurrence, yet which tend to reserve some surprises for later points in the piece.

The zone cycler consists of a relatively long sequence of terms (representing the grow cyclers) and a sequence of *zones*. Each zone is a subset of the terms in the term sequence. When cycling through the sequence, only terms included in the current zone will be emitted, and other terms will be ignored. For example, if the term sequence is [a;b;c;b;c;d;b;a;c] and the current zone is [a;b], then the emitted sequence will be [a;b;b;b;a...]. Another cycler determines which zone is currently active, and for how long.

We can bias the creation of random zones to privilege some continuity between adjacent zones, so that if the first zone contains some term x, the second zone is somewhat more likely to contain x. We can also *reserve* some terms to only appear in the last half or last third of the sequence of zones.

The zone sequence $[\{c, a\}; \{c, b\}; \{d, b, a\}; \{f, e\}; \{e, d, a\}]$ has some continuity, some surprises, and some returns after a break. Given the basic term sequence $[f; a; c; c; e; f; c; d; a; f;c; f; f; d; a; c; b; d]$, if we stay in each zone until we have emitted six shapes, then the emitted sequence is $[a; c; c; a; c; c; b; c; c; c; d; a; b; d; a; d; f; f; f; f; e; f; d; a; d; a; d]$. With the presence of a at the beginning, middle and end, the repetition of the $d; a; d; a; d$ sections (once with an intervening b), and the definite difference between the first half of the piece with a focus on c, and the second half of the piece, with e and f, this has a chance of being an engaging musical form.

The sequence above refers to six shapes: the final piece is put together by querying six grow cyclers emitting the shapes in this sequence – since these are *growing* shapes, they are different each time they recur.

Figure 2 shows and excerpt of a piece by Jack 1 ("Proud Buzzy"). Zoning is in evidence, as is growth – especially dramatically in shape B.

Extending Jack 1

All of the Jack 1 music from 2009-2013 was generated by the described meta-piece (with some slight meta-piece variations), with no runtime human interaction.

However, human intervention is theoretically possible wherever randomness occurs: instead of randomly generating cycler terms and settings, they could be set or modified by hand. Hard-coding some of the cyclers could allow more thorough and controlled exploration and mapping of smaller spaces of musical possibility.

It would also be possible to use the cycler concept to create different meta-pieces, preserving a high degree of randomness in each instantiation, but allowing experimentation with different structures of interacting cyclers. Certainly other concepts of variation, development, and formal structuring could be developed, perhaps with deeper levels of cycler interaction. A "meta-meta-piece" might have a greater degree of randomness in the connections and interactions between cyclers.

Though we are presenting one meta-piece as a musical meta-artwork, we also offer a modular, extensible concept for the random creation of musical material with controlled complexity of structure and pattern at several levels.

The Jack 1 meta-piece described in this paper is fairly conservative with regard to the complexity of structured pattern that can be generated using cyclers. To some degree this was an aesthetic choice, but there may be a cognitive limitation on what kind of pattern complexity can be musically useful.[6]

The cycler concept admits of generating musical samples of controlled pattern complexity, which could be used to develop a testable hypotheses for psychological studies examining auditory-musical cognition of pattern (without reference to learned or enculturated musical styles).

We might start by testing whether a formal model of pattern complexity based on the complexity of the generating cycler-system seems to correlate well with cognitive complexity. To see whether the structure of a pattern is apprehensible as such, we might be tempted to try prediction games, but we suspect that the game quickly becomes too hard. Detecting when a pattern has changed (i.e. switched from one cycler system to another) might be easier.

Conclusion

Jack 1 is an original meta-music piece which emits quirky, animated output. It works by constructing a random multi-level cycler structure as a virtual generating engine.

There are many features not included in Jack 1, such as a theory of harmony or polyphony, an ability to analyse its own output, and broader facilities for variation and development.

The system presents a fairly small set of operators and combinative structures that can generate compositions of 1-3 minutes that unfold in characteristically musical manners, with themes, developments, climaxes, and contrasting secondary material. It does so without access to existing music, music theory, or statistical weighting of note distributions. One conclusion is that many of the conventional categories can be produced indirectly, as process, rather than as a model of existing music. Another bears on the nature of music. Simple concepts like shape, pattern, zone and grow can interact to yield a sense of purpose, even agency: essentially this interaction is a game, a complexification of of the simple patterns and shapes present in children's games. The experiment provides a glimpse into a new ontogeny of music.

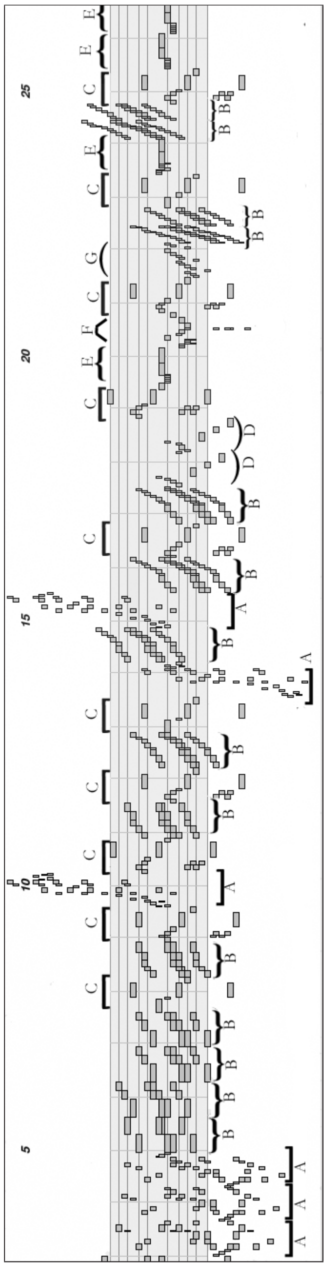

Figure 2: Jack 1 excerpt, "Proud Buzzy."

[6]Lerdahl, Fred. "Cognitive constraints on compositional systems." Contemporary Music Review 6.2 (1992): 97-121.