Cellular Automata Mapping Procedures

Peter Beyls
St Lukas Hogeschool Brussels
Computer Music Research, School of Computing, Communications and Electronics,
University of Plymouth

peter.beyls@pandora.be

Abstract

Despite of the simplicity of their construction, cellular automata are capable of very complicated behavior with very little specification. The present paper introduces an algorithmic music system incorporating a voting rule type automaton extended with energy profiles and an elaborate mapping algorithm. One core design principle is the fusion of generative, implicit behavior with explicit user designed symbolic processes, in particular, facing the mapping task.

1. Introduction

Cellular automata (CA) are mathematical models for complex natural systems containing large numbers of simple identical components with local interactions [Wolfram, 84]. They are discrete dynamical systems exhibiting a wide range of complex behavior; from on the one hand, point and cyclic behavior yielding structures of relative periodicity and, on the other hand, utterly chaotic attractors. In between, a whole range of delicate structures exist featuring areas of relative order and instability. CA prove instrumental to handle degrees of coordination and evolution; structural coherence can be studied; consistent structures may emerge from initial random states. We propose to view CA as a qualitative method to navigate areas of musical activity.

Earlier musical implementations consider one-dimensional [Beyls, 89] and two-dimensional [Miranda, 93] automata. Some programs operate on the sample level [Chareyron, 90], while other programs aim the synthesis of structures in the MIDI domain even with genetic control over the CA lookup table rule [Beyls, 95].

This paper introduces a linear CA with a voting rule, a multi pane graphic user interface and a quite complex mapping algorithm. The output of a binary CA is interpreted as a multiple value one based on the density of neighboring cells. In addition, the user specifies various control parameters. Thus, the result is a generative system where implicit behavioral wealth is influenced though not rigidly set by user interaction.
The CA displays many qualities that are reminiscent of (artificially) living systems, for instance, the synthesis of self-organizing patterns; the creation of relative order from random initial configurations. This work excludes stochastic processes -- our implementation de-emphasizes random numbers throughout -- though structures do emerge that do not appear to be totally deterministic. This may be understood as a paradox; it is both the strength and weakness of a system which excludes creative chance and has no motivation to change its own rules. In terms of musical composition, this method both accommodates experimentation with niches of potential neighboring patterns as well as exploration of the non-linearity of neighboring rules. From this it follows that there is no single best solution, the user navigates areas of relative structural opportunities.

2. Implementation

Our implementation uses two regular lattices of 6 by 64 cells; a Boolean array holding the activity of a voting-rule plus a second array holding an expanded, multi value encoded version of the former. In contrast to a rule represented as a conventional lookup table, we apply a spatial rule; it consists of a 5 by 5 matrix of weights used by a voting algorithm which works as follows. For every cell in a Boolean array, initially filled with random values, the weighted sum of the local field of 25 cells is calculated; positive weights proportionally increment a local counter of 'on' cells while zero valued cells increment a second counter. This process yields the two counter values which are finally compared; if they are equal, 0 is returned, otherwise 1 or 2 is returned according to the sign of the difference of the two counters. The two values are interpreted as 0 and 1 respectively in the next generation of the matrix.

Now, this binary matrix serves as the basis for a multi-valued one by way of a simple local operator; in effect, the opposite action of a smoothing filter as used in picture processing. The local sum of every cell yields a value between 0 and 9 using the standard 3 by 3 Von Neuman neighborhood and including the center cell. This value is used as pointer to select a color in a palette with 10 entries; binary information thus proliferates into multiple values according to local spatial densities. Figure 1 provides a snapshot of the main interface showing the two arrays at the

top. The next lower pane shows a piano roll notation of an events stream created by the application of an elaborate mapping algorithm described in paragraph 3.

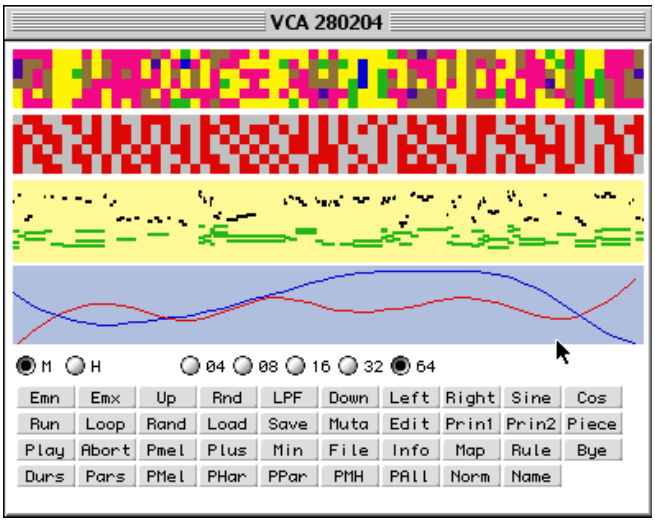

Fig 1: typical snapshot of main interface

Finally, in the next lower pane, the user can edit an energy profile; it specifies a deterministic mask quantitatively controlling the output of the mapping process though it receives a different interpretation for melody and harmony.

Fig 2: spatial rule editor specifying weights in a 5 by 5 neighborhood

The spatial rule is accessible for explicit editing in a private GUI (see fig 2) by the user; a 5 by 5 grid holds a weighting value between -5 and +5. Functions are provided to generate symmetrical rules or to mutate existing rules in order to study the effect of slight mutation.

3. Advanced mapping

3.0 Characterization

By mapping we mean the establishment of a sensible connection between two areas of activity which creates meaning to a human or machine perceiver.
Algorithmic composition implies the creation of a qualitative link between some arbitrary conceptual/generative model invented by a programmer with the elusive cultural field we call music. Some generative systems are deeply embedded in musical tradition and history while other systems only refer and obey to the virtual physics of some imaginary world.

For example, some algorithms are active on the surface level by expressing rules of Western tonality and harmonization. In contrast, some systems function according to a musical relativity admitting multiple points of view, frames of reference and operating models, music making which involves the creation of entire musical universes each built on its own unique assumptions [Rosenboom, 97].

This paper views CA as evolving universes propelled by the laws of their private physics; laws that are possibly totally alien to our understanding of the natural universe. The CA may be interpreted as sound according to the following (non exhaustive) systemic principles: (1) static system: discrete, arbitrary linear array functioning as lookup table (2) dynamic system: continuous, an invented procedural mapper of arbitrary complexity, (3) living system: a human performer, non-linear, (4) evolved cultural system: e.g. the concept of tonality, (5) psycho acoustic system: for instance, dissonance and its resolution and finally (6) natural systems: laws of form and proportion i.e. golden section.
Choosing a system has its consequences; for instance, quantization as inherent in the MIDI protocol will severely limit expressivity. Otherwise, interpretation my a human performer restricts predictability and repeatability as well.

Since a CA is considered a complex dynamical system with rich behavioral potential, the mapping algorithm should be tailored to extract a large series of features, in effect a critical, structural analysis of the CA. Thus, implicit behavior yields a series of explicit scalars to be used as musical process control variables. Note that the design of a robust mapping algorithm requires considerable manual effort and trial-and-error; one tunes and debugs the program by listening to it.

The present work attempts to introduce measures of quantity and quality in the mapping process.

By the quantity of control we mean how much influence we exercise: scalars gained from the implicit behavior of a generative system (CA here) or explicit user specified amounts.
The quality of control might be described as determined by two factors. First, the nature of the source from which to draw rules and constraints. For instance, event pitch or duration might follow, in a straightforward fashion, from the XY position plus the value of a CA cell. In this case, one fixed universe communicates a simple numeric link to

another. Otherwise, one may create numeric relationships between CA values and harmonic alternatives from a personal catalog, or one may decide to opt for micro-tonality. A fundamental cultural choice will thus steer the result.

A second factor of qualitative impact on the mapping process is the complexity of the mapping algorithm when viewed as a composite conglomerate of nested functions of arbitrary complexity. Perhaps it makes sense to match the complexity of the CA with a procedural mapping scheme of similar intricacy. Quality thus also implies degrees of understanding; a deterministic mapping function yields predictable functionality, however, the specific values cannot be anticipated by the user. This idea culminates when there is no longer any perceivable link, at this point, the user loses control because a complexity barrier is hit; the program becomes too complex to develop any further or to debug. An important conclusion is that complexity increases interestingness and decreases predictability/usability of any mapping scheme.

3.1 Melody

Mapping creates a dynamic time structure, notably melody and harmony from a static, spatial cellular structure. First the melodic component is computed from which harmony follows and finally, additional algorithms are available to derive conditional parallel voices which borrow material from melody or harmony and interweave with both. The mapping process starts by considering every time slice (6 cells) of the color encoded matrix; features are extracted resulting into nine parameters, in particular, used for the determination of pitch;

• Nv is the number of different values (1~6) and,
• Sum is the sum of all slice values (0~54, not weighted according to position),
• Pv1 is the gradient of the current sum and the sum of the previous slice
• Pv2 is the number of different values in the previous slice
• Acc is an accumulator incremented by (nv - pv2)
• Tonality set as: if (plusp (- sum pv1)) then minor else major mode
• Scale is three octave scale with root = acc
• Pitch-offset is a single cyclical list of arbitrary length set explicitly by the user.
• V1 = the value of the first element in the current slice.

Event duration is retrieved by a global counter pointing in a set of user editable duration lists (modulus the number of available lists), the specific duration list in effect is determined by Nv (which implies that there can be no more than 6 duration lists).

The energy profile for melody comes into play; the energy value (0 to 100) at every time slot simply creates a vector from all zeros to all ones indicating a note event or a rest. Finally, a note is appended only if the value of the element in the current slice does not equal V1 i.e. (or (= i 0) (not (= v(i) v1))) with (0>i>6).
Clearly, derivation of melody is a good example of a hybrid computational strategy; subsymbolic automatic behavior meets a great deal of symbolic processing.

3.2 Harmony

As outlined above, we aim a mapping method which integrates strength of control plus, more involved, a qualitative specifier. Harmony is partially computed using a simple constraint satisfaction method in combination with an harmonic energy profile specified by the user; an energy profile can be an arbitrary user drawn shape or, alternatively, it can be a wave-like pattern i.e. values computed from a trigonometric function. In either case, the profile translates to a one dimensional lookup table holding 64 quantized values. Harmony follows from a critical analysis of individual CA slices in time as well as the (dis)similarity of any two neighboring slices.

The energy profile for harmony operates as follows; the energy value (0 to 100) at every time slot is scaled to a value between 0 and 63, this becomes a pointer in a list holding 64 configurations of 6 bits. However, this list is sorted according to density of ones; it thus contains sublists with total number of occurrences of (1 6 15 20 15 6 1) in that specific order for a summed configuration from 0 to 6. For example, there are both 15 lists that hold 2 ‘on’ bits as well as 15 lists that hold 4 ‘on’ bits. The rationale here is that high energy values retrieves ‘many on bits’. The density of the bit vector will correspond to the selection of specific harmonic functions of any given chord.
Now, a list of global constraints may influence harmonic continuity. Parameters include: (1) the number of pitch classes required to be common in the melodic segment and the chord, (2) instructions for imposed root movement, (3) the required tonal intersection size between any two consecutive chords, (4) the minimum and maximum harmonic tension, considering relative dissonance of all chordal intervals to be contributed. At every instance, a collection of harmonic alternatives is computed from the pool of all potential ones which is based on the following tonalities; xm7, x7 and x7M and their variations: xm5b7, xm7M and x5#7M. The algorithm is purely first order, if it works itself into an impasse, either backtracking occurs or the selective pressure of a random constraint is gradually loosened until a fit solution is found.
Besides the fact that one makes abstraction of conventional rules of voice leading and voicing, the number of

permutations to articulate a list holding up to 6 chordal components over a given time slot is astronomical. It seems obvious to turn to genetic techniques to tackle a search space this size. We actually implemented such a genetic workbench holding 64 arrays six by six arrays, each array representing an initial random articulation. The idea was to view the array as a genotype specifying the articulation of a given chord.

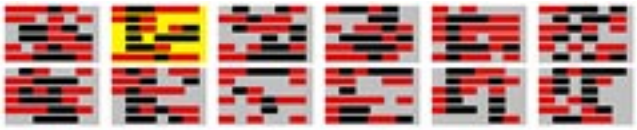

Fig 3. collection of genotypes representing articulations of a single 11th chord

The user selects relatively fit genotypes and breeds the next generation using the standard operators of cross-over and mutation. The user identifies families of interesting patterns by interactively exploring the state space through the selection - evaluation cycle. Evaluation here implies explicit subjective action by the user which illustrates a major weakness of the approach; one has to listen to an individual to evaluate. It takes too much time and causes too much human fatigue to attribute individual fitnesses; the idea was not further exploited though neural networks that learn user preference profiles which can be used in later sessions could be a solution to address this fitness bottleneck. Note that genetic methods have appeared in much music research recently -- see [Burton and Vladimirova, 99] for an excellent overview.

3.3 Parallel voices

Optional parallel voices are constructed using a 3-part parallel rule: two lookup tables holding intervallic offsets from a parent voice, for both positive and negative source intervals, and a cyclic grouping vector, specifying how many melodic source events are taken as a group to compute the duration of the parallel event. Negative entries in this vector denote rests. In addition, a MIDI channel number is given to specify which melodic events are eligible to function as source events. The creation of parallel voices requires the source melody to be segmented; consecutive events are grouped into isolated sublists according to a critical parameter specifying the maximum delay between any two events intended to be perceived as contiguous.
The resulting composite object may be saved as a MIDI file or imported for further treatment or analysis in a companion workbench documented in [Beyls, 03]. All programs are implemented in Macintosh Common Lisp and use MIDI functionality provided by Common Music [Taube,03].

4. Conclusion

CA are attractive models for musical experimentation because they support a wide variety of complex yet coherent behavior with great computational economy. Work with CA is a good example where implicit structural wealth receives musical meaning from a mapping process shaped by explicit design -- in effect a compositional method where automatic processes and direct instruction are in delicate balance. In addition, our implementation integrates subsymbolic automacy with considerable symbolic computing, most discernible in the mapping algorithm.
Finally, a CA is also a system which favors serendipity i.e. finding interesting patterns without looking for them explicitly. After all, there is no single best solution; the user navigates areas of relative interestingness and probes a given area to retrieve exemplary material.

References

- Beyls, P. (1995) *The exploration of musical space by way of Genetic Algorithms,* ISEA95 Conference, Universite de Quebec, Montreal, Canada

- Beyls, P. (1989) *The musical universe of cellular automata,* Proceedings of the International Computer Music Conference, Columbus, Ohio 1989

- Beyls, P. (2003) *Selectionist musical automata, Integrating explicit instruction and evolutionary algorithms,* IX Brazilian Symposium on Computer Music, Campinas

- Burton and Vladimirova, (1999*) Generation of musical sequences with genetic techniques*. Computer Music Journal, 23:4, MIT Press

- Chareyron, J, (1990*) Digital Synthesis of Self-Modifying Waveforms by Means of Linear Automata*, Computer Music Journal, 14:4, MIT Press

- Miranda, ER. (1993*) Cellular Automata Music; an interdisciplinary music project.* Interface, Journal of new music research, 22:1

- Rosenboom, D. (1997*) Propositional Music: on emergent properties in morphogenesis and the evolution of music. Part II; Imponderable forms and compositional methods*. Leonardo Music Journal nr. 7

- Taube, H. (2003) *Notes from the metalevel,* Zwets and Zeitlinger Publishers, Holland

- Wolfram, S. (1984) “Universality and complexity in cellular automata”, in Physica D, volume 10, pp. 1-35