Musical Sonification of Cellular Automata

The concept of the cellular automation ("CA") dates back to the earliest speculations on computing machinery. The idea was developed by Von Neuman [1966] and Stephen Wolfram continues to explore the field extensively [1994], [2002]. Cellular Automata have served as a basic theoretical tool in computing and periodically come to the public consciousness in forms such as Conway's "Game of Life" [Gardner 1970]. There is an excellent popular explanation in David Peak and Michael Frame's book *Chaos Under Control* [1994], and Wolfram has just published an exhaustive treatment in *A new Kind of Science* [2002]. Eduardo Reck Miranda has applied the principles of cellular automata to music in his CAMUS program. [2001]

One Dimensional Cellular Automata

The mechanism of cellular automation is simple. Starting with an array of cells which may each take a value of 0 or 1, calculate a new array in which the state of each cell is determined by the previous condition of itself and its neighboring cells using a short list of rules.

For a one dimensional array, a rule may be expressed as a set of binary numbers, representing the left neighbor, the state of the cell under examination, and the right neighbor. As an example {0 1 0}, indicates that the cell is active but its neighbors are not. The rules are taken as OR controls. If any of the rules matches the state of the current generation, the new state of the cell is 1. Figure 1 shows the first 15 arrays generated by applying the rules {1 0 0} and {0 0 1} to a single cell.

Fig 1.

Successive generations of one dimensional arrays are usually shown as a history running from left to right or top to bottom. In figure 1 it can be seen that the first cell engendered two cells, each of which generated a child, and each of those was in a position to engender yet another pair. This expansion will continue until the corners hit the end of the array space, in which case the next array will be empty.

Figure 2 shows what occurs when the rule {0 1 0} is added to the list.

Fig 2.

Varying the rules and the initial seeds can create diverse shapes, all of which share the triangular flavor with different fill patterns. Increasing the number of cells evaluated each generation can change the shape of the triangle and rate of growth, as well as provide more variations on the internal structures. When the rules are symmetrical, the total number of active cells in the array follows cyclical

numeric progressions, such as 1 2 2 4 2 4 4 8 2 4 4 8 for figure 1 or 1 3 2 6 2 6 4 12 2 6 4 12 for figure 2. As we will see later, these patterns can provide frameworks for interesting sonification strategies.

Rule checking may include more than the immediate neighbors and may be extended back more than one generation in a manner analogous to Markov processes. The result of a third order computation is shown in figure 3. The rules were { 1 0 0 } and {0 0 1} in generation –1 and the same in generation –3.

Figure 3.

This class of automaton grows until the width of the array is filled with a pattern that does not answer to any current rule. At the edge of the array, the programmer may choose to either terminate rule checking (all references to cells outside of the array return 0), or wrap checking to the opposite edge. (In an array of 128 points, reference to cell [128] returns the state of cell [0]). If wrapping is used the size, particularly whether even or odd or a power of two, will have strong influence over the appearance after wrapping.

A third option is reflection, where reference to cell[128] returns cell[127]. This is equivalent to wrapping when the patterns are symmetrical, but produces quite different results when the rules or seeds are unbalanced.

Two Dimensional Cellular Automata

John Conway's Game of Life is based on two-dimensional cellular automata. Two-dimensional CAs can be controlled by point specific rules in a manner

similar to the one-dimensional variety. This requires sets of 9 members which yield 512 possibilities. A large array may evaluate slowly, even on modern machines. The rules are usually simplified to considering the total number of adjacent cells currently active and the current state of the cell under examination. The definition of adjacency may or may not include corner neighbors. In the Game of Life, all eight neighbors are counted, and the rules for activation are {3 0}, {2 1} and {3 1}. Because of the lack of directionality in the rules, symmetrical seed patterns develop into other symmetrical patterns, but inevitably the patterns devolve into stable or cyclically repeating groupings. Figure 4 illustrates this. Starting from a square seed, the pattern proceeds through a diamond, a partially filled diamond, a larger square, a group of four crosses, then four separated 3 cell lines, which will cycle endlessly between vertical and horizontal. Asymmetric patterns can be constructed which grow indefinitely, and the design of such patterns has inspired quite a few papers. Some seeds will not grow at all. In particular, an isolated cell will become extinct on the next iteration.

Figure 4.

With the addition of the rule {1 0}, symmetrical patterns will grow quite large and complex. In an infinite data space, they would grow indefinitely, but even in bounded arrays, they take hundreds of generations to fall into obvious cycles. This additional rule leads to chaotic productions where slight differences in the seed pattern will create very different histories and settle on different attractors.

Figure 5 shows how a square seed develops. In a 16 by 16 space, this production requires 240 steps before it falls into a 17 step repeating sequence.

Figure 5.

Figure 6 shows the first patterns from an X-shaped seed. On a 16 by 16 field it grows through 145 stages before it begins to cycle in a 6 step pattern.

Figure 6.

In an 18 by 18 field, the same seed develops over 216 steps then locks into the stable shape shown in Figure 7.

Figure 7.

In larger fields, the number of steps to a repeating cycle tends to increase, but any bounded array must fall eventually into cyclical behavior (Wolfram 2002). Odd sized arrays tend to give longer cycles. An X-shaped seed in a 17 by 17 field does not repeat after 1000 iterations.

The patterns shown so far are radially symmetrical. They need not be. A bilateral seed will produce bilateral patterns and an asymmetrical seed produces asymmetrical patterns. Figure 8 shows a typical bilateral pattern.

Figure 8.

To maintain symmetry after the array fills, seeds in an odd-sized array should have an odd width and those in an even-sized array an even width.

Perturbation

Even if the patterns inevitably settle into cyclical behavior, there is a simple solution to maintain variety: change the rules. Since the progression of shapes with the addition of the {1 0} rule is different from that of Conway's original set, making this rule conditional on outside factors encourages a unique progression with each run. Within the context of a musical performance, the activation of the rule may be a function of time or performer input. In the latter case, the input may be deliberate control or side effects of musical actions. In stand-alone installations, the rule may be controlled randomly or triggered by the detection of cyclic behavior in the output. If multiple patterns are used, they may interact in various ways.

Normally, a programmer would ensure that rule changes only occur in the time between generations, as switching during calculation of the array can result in skewed patterns. In fact, these represent another source of interest, and can produce forms that gradually become asymmetrical.

Added variety may also be generated by injection of additional seeds or erasure of active cells. Changing the dimensions of the array either progressively or suddenly will stir the pot in dramatic ways. Again, these actions may be performer instigated or automated.

An endgame can be forced by deleting rules. This will usually cause the patterns to thin out and disappear.

Sonification Strategies

With the rich variety of cellular automatons to generate data fields, it is rewarding to explore triggering of sounds by the activation of cells. The pattern in figure 9 is the 100th generation of a 127 by 127 array. The initial seed was a single centrally placed cell, which was repeated 5 times in the first 10 generations. The {1 0} rule was initially on and shut off at generation 50.

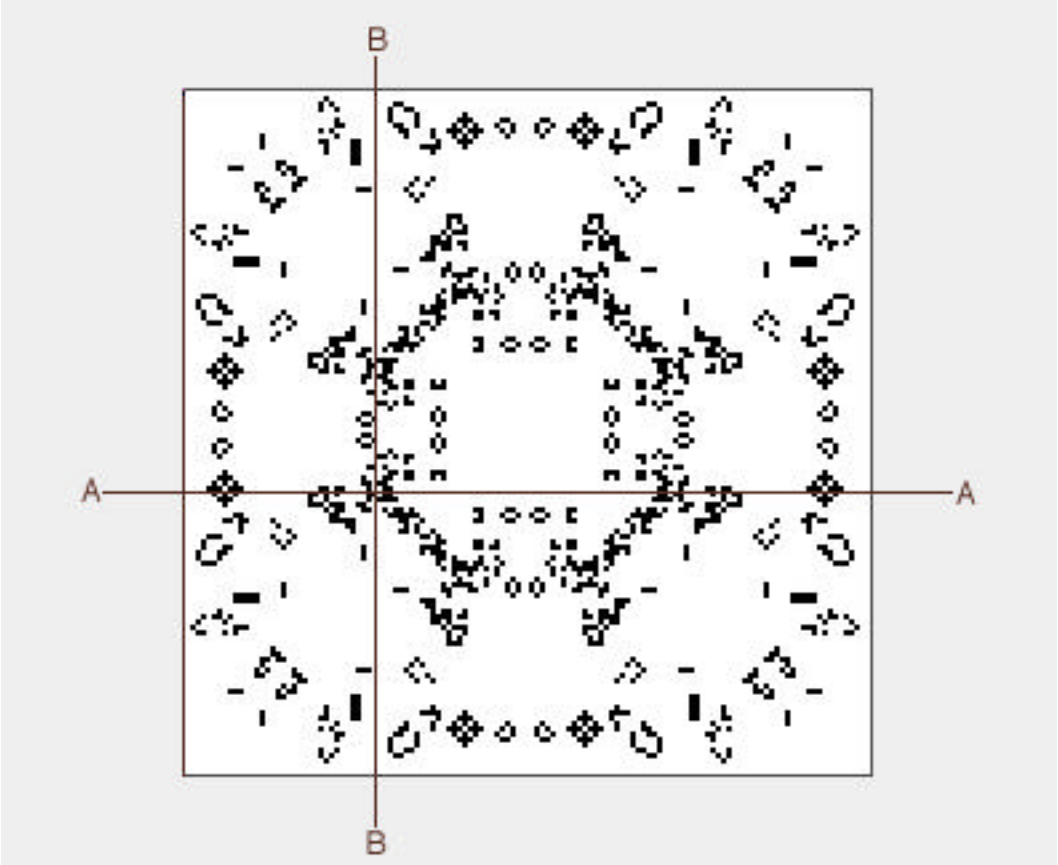

Figure 9.

There are several possibilities for sonification. Taken as a static structure, the dark points along the line AA will produce a set of the type { 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 …} with 127 members. This can be converted to a list of the active positions. The above would yield [8, 11, 14…..]. This may be treated as a chord, a row or a

rhythmic specification. The sum of the members of the set provides a single number that can be used for control.

The line BB produces a different set. The fact that BB is oriented at right angles to AA is not significant in this example, but in non-symmetrical patterns a different group of sets occurs on each axis. Sweeping the line BB along the axis AA will produce copious amounts of data. The sums can be played as a melody, or the derived lists may be used as a harmonic progression.

In the dynamic case, the patterns grow and sets are taken periodically. Sets taken from the same location would then reflect the growth of the structure. Alternatively, a sampling of sets or a sweep may be performed. If each sampling position is associated with a voice, a quasi canonical effect may be achieved as the initial form grows to reach the outer ranges.

In yet another approach, scattered points may be probed on each iteration. Since regions near the center tend to thin out as the growth approaches the edge (and then fill in again), point sampling often results in interesting formal structures. In particular, such points may be linked to several simultaneous patterns for multi-dimensional constructions.

Contrapuntal systems may be constructed by separately considering sub-regions of the array.

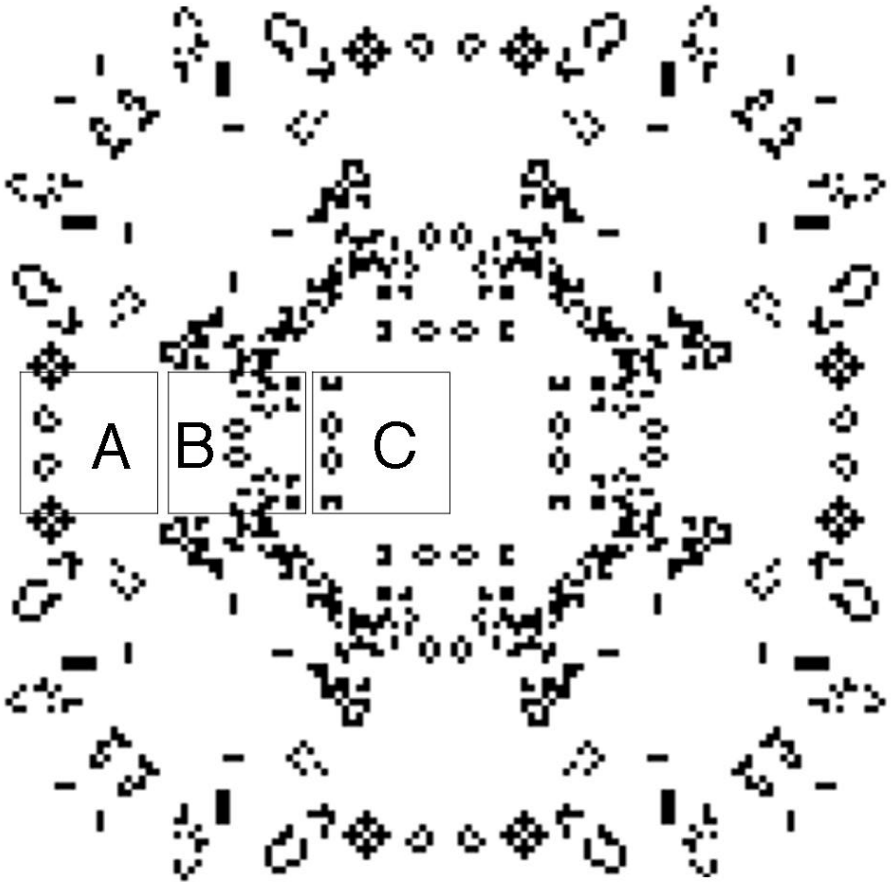

Figure 10.

In figure 10 the array has been partially subdivided. Taking the total number of active cells in each region will give slowly changing values that reflect the development of the array as a whole. Extracting short sets from each region will create motifs that are loosely related, but not exact repeats. For instance, the region C will become active before B or A. When the array grows into B, the history of the sets will be similar to what happened in C, but not a literal repeat. The activity in A will commence last, and bear more resemblance to B than to C. Once the patterns reaches the edge, reflected activity will impinge on A before B or C.

Transformation

The raw data from the sets may be satisfactory in some contexts, but more likely the composer will massage the data toward some tonal scheme. In this case, sieve or funneling techniques are very useful. In a sieve, values that are not on a preselected list are rejected, resulting in thinner data. In a funnel, if a value is not on the list, it is rounded to the nearest desirable point. Sieve and funnel operations are simplified if the data is converted to pitch class before the processing function. Octave information may be kept and restored or replaced by some other registeral component.

Another useful process is to look for edges or centers of clusters in the conversion of sets to data lists. With this system, a set { 0 1 1 0 0 0 1 1 1 0…} would yield [1, 6,…..] in the converted list. Each cluster produces one value, which gives a simplification of the overall pattern while retaining the essential shape.

Another way to reduce the data is to recognize the symmetry of the sets and convert only part of them. Conversion from the beginning of sets will give asymmetrical rows, whereas conversion of a subset centered in the domain will maintain symmetry.

An Example

The author's composition "Patterns" demonstrates a basic sonification technique. The work for performer and MIDI instrumentation with projected graphics. The software is built on the Llife Max object, which maintains a 128 by 128 cellular array and displays it as a sprite in a Max graphic window. Llife is part of the Lobjects for Max [Elsea 1996], a package of external objects that add high level operations to the core program.

Llife displays feature 24 bit color and can be layered in a single graphics window. Rules and seeds are set with simple commands and new generations are calculated in response to a bang message. A total value is output on each cycle and the current state of the array can be queried by column, row or point.

For Patterns, only three rules are active: {1 0}, {3 0} and {3 1}. From a single dot, the form grows rapidly, as shown in figure 11.

Figure 11.

Sonifcation is quite simple. The bottom half of the figure is scanned, and the active pixels are mapped to notes which are played by assorted percussion sounds. Figure 12. gives an approximation of the rhythms resulting from the opening six stages.

Figure 12.

With no interference, this piece is absolutely determinate. The performer is instructed to occasionally add a center dot, which will switch the piece along one of 64! paths. Eventually patterns such as figure 13 appear. Since the figure will maintain its symmetry and tends to clump into three part forms, the sense of rhythmic integrity is maintained.

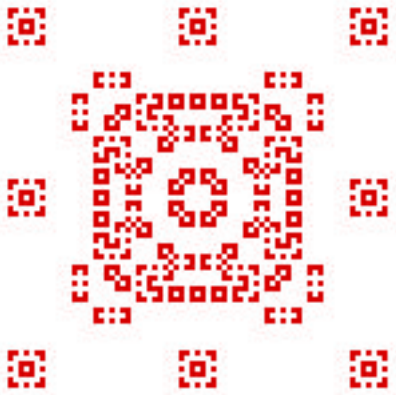

Figure 13.

The software will turn off rules {1 0} at generation 59, so the work will end with a clearing of pixels.

Spectral Sonification of Three color CAs

Wolfram [2002] presents an extended analysis of three color cellular automata. These are one dimensional arrays in which cells may take one of three states, depending on the sum of the states of it and its neighbors in the previous generation. Most of these give more complex versions of the pyramids of the 2 state version (which are exhaustively explored by Wolfram) but a few are self limiting and musically quite interesting. These produce very tight patterns that lend themselves to spectral sonification, where each cell is mapped to frequency of grains in a texture. In particular number 1599 [Wolfram's classification, pp 69, 70] has a fascinating structure.

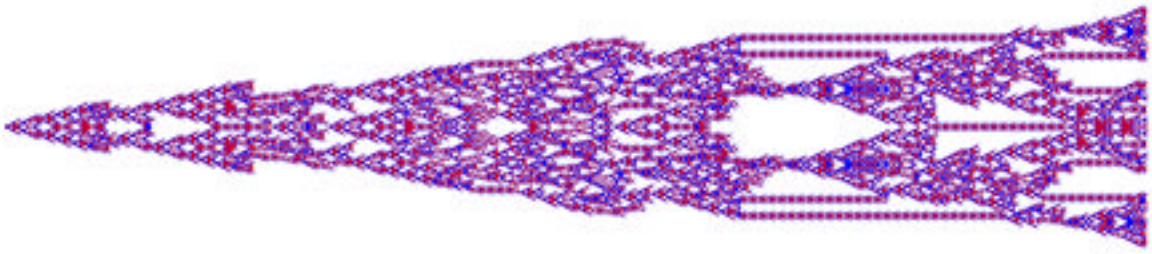

Figure 14.

The beginning of type 1599 is shown in figure 14. The complex expansion continues for nearly 3000 generations, then the figure begins to contract, leaving a few narrow "streamers" such as those visible near the lower right of the figure.

Sonification of a structure as complex as figure 14 can benefit from some data reduction. At its height, the figure covers nearly 800 cells top to bottom, which would require frequency spacing on the order of 10 cents. This makes the subtle gestures derived from the angular edges of the inner structure very hard to discern. If only the top half of the pattern is used, a 25 cent spacing will fit in the audible range and give an impression of rapid motion within the cloud of sound.

References

Elsea, P. [1996, rev 2003] Lobjects, internet distribution, available at ftp://arts.ucsc.edu/pub/ems/lobjects

Gardner, M. 1970. *Mathematical Games* . Scientific American, October.

Miranda, Eduardo Reck 2001. *Composing Music With Computers*. Oxford, Focal Press

Peak, D. and Frame, M. 1994 Chaos Under Control: *The art and Science of Complexity*. New York: W.H.Freeman

Von Neuman, J. 1966 *Theory of Self-Reproducing Automata.* Urbana: University of Illinois Press

Wolfram, S. *Cellular Automata and Complexity: Collected Papers.* New York: Addison-Wesley, 1994

Wolfram, S. *A New Kind Of Science.* Champaign IL, USA: Wolfram Media, 2002