Hacking Cellular Automata: An Approach to Sound Synthesis

Yota Morimoto
University of Birmingham, UK
http://yota.tehis.net/

In this paper, I present my implementations of cellular automata in sound synthesis (UGens in SuperCollider). A general explanation of the system is given, followed by a review of historical applications in music. It then gives descriptions of the actual hacking/implementation and conclusions drawn.

Keywords

cellular automata, sound synthesis

1 Introduction

Musical applications of cellular automata are very common today. Nonetheless, little attention has been drawn to the seemingly necessary adaptation involved when using such systems to create musically plausible results. The motivation of this paper is to describing my compositional considerations in applying cellular automata, rather than postulating the computational elegance they present.

1.1 Cellular automata

A cellular automaton (CA) is considered as a mathematical model of self-replicating machines. Practically, it is a system which produces sequences of discrete states encoded in a lattice of cells in one or more dimensions. For the sake of brevity, the figure shown is a one-dimensional binary state cellular automaton with nearest neighboring configuration [Fig.1].

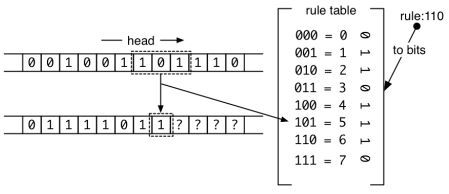

Figure 1: CA transition mechanism

Each cell can hold its state often represented as integers or real numbers. A new lattice is generated based on the states of the past lattice. To determine the value of each cell in a new lattice, contiguous sections of cells from the previous lattice are examined (indicated in Fig.1 as the reading 'head'). A rule table specifies all possible transitions with the neighboring configurations. In this case, combination of three binary cells $2^3 = 8$ is the amount of possible transitions the rule table must specify. The rule table is simply an array of 8 bits, that is here "01101110," and the integers represented by the three neighboring cells point to the indices of the table to determine the new state of the cell (in the case of Fig.1, the examined three bits "101" become integer 5 that points to the 6th element of the rule table.) Instead of using the binary notation "01101110," integer representations of such 8 bits are conventionally used. The range of the rule is thus, $2^8 = 256$ (0-255), and in the case of Fig.1, the rule table "01101110" can be denoted as 110.

Although cellular automata do not prescribe any particular model of dynamical system, they provide alternative and pragmatic ways to approach the problem of describing and analyzing complex systems. The simplicity of computational descriptions in CAs is instrumental to the understanding of the mechanisms responsible for the generation of complexity and a study shows that such systems are capable of emulating other systems with any degree of periodicity, including chaos [Wol02, pp.675-691].

2 Musical applications

As complex yet controllable systems, CAs have been a stimulating source of ideas to many artists including musicians.

Motivated by CA's relevance to fluid turbulence, Iannis Xenakis in his orchestral work *Horos* (1986) used a one-dimensional system (*linear automaton*) to produce harmonic progressions and orchestration. At first, musical scales are constructed using his *sieve theory*, and CAs determine the unfolding of the scales (i.e. the temporal order of the scales, and

their assignments to groups of instruments). As is commonly observed in Xenakis' other approaches, the utilization involved numerous arbitrary alterations of the system's output [Sol05].

Peter Beyls has been using CAs for algorithmic composition and performance. He developed a system in which the composer or the performer could stir the cells to preferred directions realtime, using a mouse pointer and/or a physical controller, and has been extended the model's formulations with an application of a feedback mechanism and multi-level automata [Bey89, pp.40-41] [Bey91, p.257].

The intriguing visual aspects of CAs were also incorporated in the multi-media installation work presented at STEIM: Joel Ryan used a linear automaton to control a FM synthesizer and the Fairlight video processor [Rya08].

More recently, application to control granular synthesis has been implemented in the commercial software synthesizer Chaosynth [Mir01], and application to sound synthesis (whose implementation draws similarity to the Karplus-Strong physical modeling algorithm) has been reported in detail by Jacques Chareyron [Cha90].

The approaches presented are diverse and practical. Notably, those of the successful ones often deviate from the standard CA implementation, and this has also been a key to my application in sound synthesis.

3 Implementation

CA0, CA1, CA2[1] are UGens for the computer music language SuperCollider [McC02]. As the development of the sound synthesis models has been rather experimental, SuperCollider's realtime controllability in combination with the other built-in UGens was crucial.

3.1 CA0

First, I decided to use the simplest system of all, the *elementary automaton* (binary states linear automaton of nearest neighboring configuration), to study if further complicated systems are necessary. In the elementary automaton, since cells's states only vary in two ways, either 0 or 1, sample values are generated by computing the mean density of the living cells. If the lattice size is 100 and there are 50 living cells represented by the state 1, averaging the array of 100 binary digits yields 0.5 (that is 50/100) which represents a single instantaneous sound pressure level. In this way, the sample values always fall within the range of 0–1 independent from the lattice size. Encoding the whole lattice to obtain a single sample value is computationally rather expensive in comparison to Chareyron's implementation in which a single cell corresponds to one sample value.

3.2 CA1

In the CA1 UGen, different implementation which translates the array of binary digits to integers was investigated. This implementation differs from the mean density model discussed above in that every pattern of binary digits will yield identical sample values and thus, exploits the complete dynamics of the system. Contrary to my expectation, the basic sonic characteristics were similar between the two different implementations. Based on the fact that the mean density model produces undesirable DC offset (in addition to favoring the more detailed sonority of the CA1), I decided to discard CA0 from the official release (subsection updated in 2012).

3.3 CA2

I have also created another cellular automaton UGen with more complicated underlying rule (two neighboring configuration, named CA2), but it had not led to much greater complexity in the overall behavior. The amount of transitions the rule specifies become very large, since $2^5 = 32$ is the possible amount of combinations of two adjacent cell configurations (five cells in total), with 32 bits there are $2^{32} = 4294967296$ possible rules, which makes it impractical to study.

3.4 Hacking CA

The two unconventional features which the UGens implement are the *dynamic control of the lattice size*, and the *reseeding* function, both of which perturb the running system.

While experimenting with constructions of different lattice size, its general relation to the sonic behavior has been observed. With a limited amount of interacting cells, the system tends to act in repetitive ways resulting in pitched sounds, and the more the cells get involved, the complex and noisy the sounds become. Changing the lattice size rapidly results in highly articulated sounds whose internal organization varies quickly between the extremes, with many gaps left in between. This depen-

[1]The UGens have been released open-source under GPL in 2012, and are obtainable from; https://github.com/yotamorimoto/ca4sc

dency of cells's activity on the lattice size turned out useful for controlling the sonic property of the system, and has been adopted to the three UGens.

Out of the 256 possible rules, there are roughly only 10% that continue evolving over time, and the rest either quickly die away towards a zero population or maintain a fixed amount of living cells, in both cases no sound is produced. The reseeding function was initially devised to utilize those "unfortunate" rules. By reseeding (i.e. refilling the lattice), it became possible to stimulate the system when attracted to undesirable fixed-points. Despite the quick decay some of the rules produce, it became possible to form a continuum when the system with such rules are exited sufciently dense. Timbre is determined by the seed, therefore, the function also allows exploration of the timbral spaces other rules specify.

The two hacking functionalities embedded in the UGens are somewhat parallel to what Christopher Ariza proposed as "Automata Bending" in algorithmic composition. Borrowing the term from circuit-bending (creative short-circuiting of electronic devices), he states that in artistic applications of complex systems "there is no mandate to use 'pure' implementations," justifying the utilization of cellular automata by adding external functionalities similar to the kind I have described. Additionally, with some other bending technics he devised (cell state mutation and probabilistic rules), he realized "more-flexible controls to shape and even direct CA evolution." [Ari07]

4 Conclusions

Adding complexity to the underlying rules for a system did not ultimately lead to more interesting musical result. As studied by Wolfram:

> "it seems that all the essential ingredients needed to produce even the most complex behavior already exist in elementary rules [...] using more complicated rules may be convenient if one wants, say, to reproduce the details of particular natural systems, but it does not add fundamentally new features."
>
> [Wol02, p.62]

Such minor details remain out of scope so far, however, in light of the affinity of cellular automata to fluid dynamics, they are still suggestive for sound synthesis of physical modeling nature.

References

Christopher Ariza. Automata Bending: Application of Dynamic Mutation and Dynamic Rules in Modular One-Dimensional Cellular Automata. *Computer Music Journal*, 31(1):29–49, 2007.

Peter Beyls. The Musical Universe of Cellular Automata. *ICMC Proceedings*, pages 34–41, 1989.

Peter Beyls. Self-organizing control structures using multiple cellular automata. *ICMC Proceedings*, pages 254–257, 1991. hardcopy.

Jacques Chareyron. Digital Synthesis of Self-modifying Waveforms by Means of Linear Automata. *Computer Music Journal*, 14(4):25–41, 1990. hardcopy.

James McCartney. Rethinking the Computer Music Language: Super Collider. *Computer Music Journal*, 26(4):61–68, 2002.

Eduardo Reck Miranda. Evolving Cellular Automata Music: From Sound Synthesis to Composition. *Proceedings of the Workshop on Artificial Life Models for Musical Applications*, 2001.

Joel Ryan. about LINA. at personal conversation, 2008.

Makis Solomos. Cellular Automata in Xenakis' Music. Theory and Practice. *Definitive Proceedings of the International Symposium Iannis Xenakis*, 2005.

Stephen Wolfram. *A New Kind of Science*. Wolfram Media, 2002.