Interface

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/nnmr19

Cellular Automata Music: An Interdisciplinary Project

Eduardo Reck Miranda [a]

[a] Faculty of Music , University of Edinburgh , Alison House 12 Nicolson Square, Edinburgh, EH8 9DF, Scotland, UK E-mail:

Published online: 03 Jun 2008.

To cite this article: Eduardo Reck Miranda (1993) Cellular Automata Music: An Interdisciplinary Project, Interface, 22:1, 3-21, DOI: 10.1080/09298219308570616

To link to this article: http://dx.doi.org/10.1080/09298219308570616

Interface, Vol. 22 (1993), pp. 3-21

0303-3902/93/2201-0003$25.00

Cellular Automata Music: An Interdisciplinary Project

Eduardo Reck Miranda

ABSTRACT

Researchers from many branches of scientific knowledge have been attracted by the organizational principles found in music. Also music has been an interesting domain for the application of new discoveries in science, inviting people to combine artistic creativity with scientific methods. A new environment has emerged in the domain of musical composition; what is now called *computer music*. This project is an attempt to identify correlations among different disciplines such as biology, crystallography, computing and fluid mechanics in order to investigate the possibility of composing music inspired by a framework of interdisciplinary knowledge. We have selected cellular automata to play the central role on our research due to the fact that they have been used to model a wide range of scientific phenomena, such as chemical reactions, dynamic systems, chaotic systems, genetics, crystal growth, and so forth. This paper describes a prototype implementation of an experimental system for cellular automata music composition. Musical aspects related to the system are also discussed.

The system is composed of three programs: CAMUS (Cellular Automata MUsic), CAMPLAY (Cellular Automata Music PLAYer) and PROCAMUS (PROlog and Cellular Automata MUSic).

CAMPLAY is a cellular automata music player, to perform score files processed by CAMUS and PROCAMUS. PROCAMUS is an expert system to manipulate musical data generated by CAMUS. Finally, CAMUS is the program responsible for the composition of cellular automata music itself (Fig. 1).

This paper introduces the fundamentals of CAMUS. CAMPLAY's details are not relevant here since it comprises a subset of CAMUS' subroutines. PROCAMUS is explained elsewhere (Miranda, 1991) and is not the scope of this paper.

CELLULAR AUTOMATA. AN OVERVIEW

Cellular automata were originally introduced in 1963 by John von Neumann and Ulam (Cood, 1968). They wished to create an automaton to deal with the control, informational, and logical aspects of artificial man-made systems (like computers) and natural systems (biological systems and physical processes).

Manuscript received October 15, 1991.

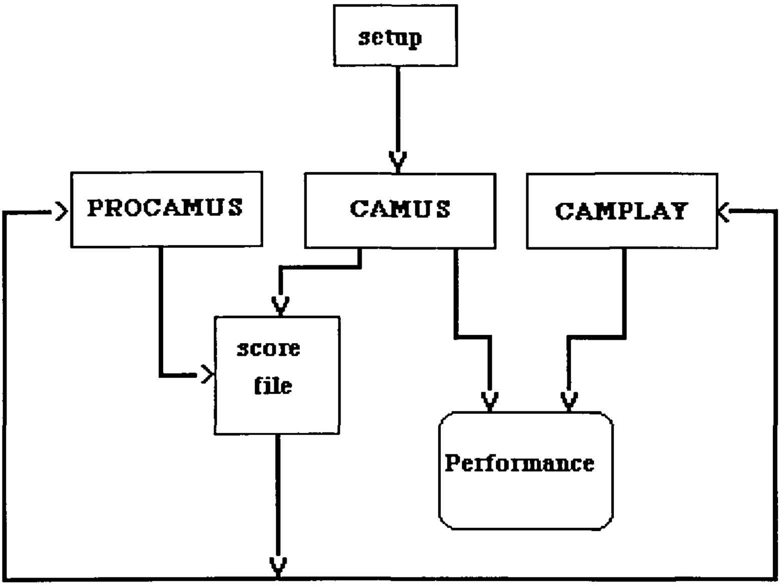


Fig. 1. Cellular automata music system.

Cellular automata are mathematical models for complex systems containing a large number of simple identical components. They consist of a lattice of cells in which each cell can take on a finite number of possible values, or states (Wolfram, 1984). The values of the cells evolve synchronously in discrete time steps according to identical rules called transition rules. The value of a particular cell is determined by the previous values of a neighbourhood of cells around it.

For musical purposes cellular automata may be widely used since music may be composed based on pattern propagation and formal manipulation of its parameters. Musicians might try to find out new compositional techniques based on cellular automata.

From many different types of cellular automata algorithms in existence, two were selected for use in this project in an attempt to investigate their musical possibilities: the *Game of life*, invented by John Horton Conway (a University of Cambridge mathematician), and *Demon cyclic space* discovered by David Griffeath of the University of Wisconsin at Madison (Dewdney, 1985).

The game of life cellular automaton

Game of life, or simply *life*, is an infinite two-dimensional lattice of squared cells whose states (*0* or *1*) are influenced by the states of neighbouring cells. Time is also discrete and from one tick of a virtual clock to the next each cell is either alive (*1*) or dead (*0*) depending on a set of very simple rules:

i. if a cell is dead at time t, it comes alive at time $t+1$ if, and only if, exactly three of its eight neighbours are alive at time t;

ii. if a cell is alive at time t, it dies at time $t+1$ if, and only if, fewer than two or more than three neighbours are alive at time t.

With this rule acting everywhere on *life's* lattice, an initial configuration of live cells may either grow interminably, fall into a cyclic pattern, or eventually die off.

CAMUS' implementation enables user specified rules other than Conway's original rule.

The demon cyclic space cellular automaton

Certain mathematical systems, although described by simple algorithms, can generate miniature universes of incredible complexity.

The cellular automaton discovered by D. Griffeath provides a good example of such a miniature universe. Initialized to a random state, it always ends up with strange crystalline growths (Fig. 2) that remind one strongly of primitive forms of life.

Demon relies on an "absurdly simple rule to produce striking phenomena of scientific interest and great beauty" (Dewdney, 1989). The rule is based on the numbering of n states from 0 to $n-1$. A cell that happens to be in state k at one tick of the clock must dominate any adjacent cells by the next tick that are in state $k-1$. Domination is indicated by a change of state of the adjacent cell, from $k-1$ to k. The rule resembles a natural chain: a cell in state 2 can dominate a cell in state 1 even if the latter is dominating a cell in state 0. But it is a cyclic space, so the chain has no end because a cell in state 0 dominates neighbouring cells that are in state $n-1$.

On the basis of that simple rule, demon can turn a random distribution of

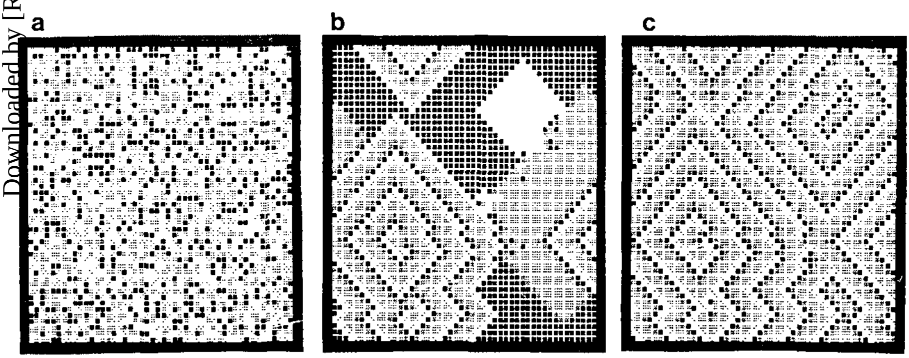


Fig. 2. Initialized to a random state (a), the cellular automaton discovered by D. Griffeath always ends up with crystaline growth (c).

colours into stable, angular spirals. CAMUS' implementation of demon uses the following algorithm:

```
repeat until stop command
for i = 1 to 48
    for j = 1 to 48
        for each neighbour (k,1) of (i,j)
            if old (k,1) = old (i,j)+1
                do new (i,j) = old (k,1)
    end loop j
end loop i
```

CELLULAR AUTOMATA MUSIC FUNDAMENTALS

Before we begin to discuss cellular automata music itself, the fundamental concepts should be established.

Rather than propose a simple, arbitrary "cell-to-pitch" map, CAMUS attempts to identify correlations between concepts from biology, crystallography and computing in order to define compositional properties.

Geometric representation of triads by means of *von Neumann* Musical Space

Consider a set of triads, where the term "triad" refers to sequence of three elements from a tune-system set. The tune-system set may be regarded as a discrete framework around which the musical events take place. Its elements – pitches – can be identified as being on a lattice. The tune-system adopted to carry out this study is the 12 tone equal-tempered system.

The set of all triads having the same interval series is regarded as a transpositionally equivalent class of triads. Any triad in a transpositionally equivalent class may be transformed into another by an operation known as ordered transposition (Forte, 1973).

The above concept is very important because it suggests a mathematical model in two dimensions to represent triads. Each dimension of this model corresponds to a specific order position in the interval series, and is quantified to represent the full range of intervals that could span a pair of pitches (Fig. 3 and Fig. 4). This model for music representation is called *von Neumann Musical Space* (Miranda, 1991).

Since the above ordered space cannot be infinitely extensible it ought to be modelled as a squared lattice drawn onto a torus (Fig. 5).

A Cartesian coordinate of a cellular automata cell can be viewed as an address to a point into a *von Neumann* musical space.

Consider the finite automaton

$$(SG, sg_0, f)$$

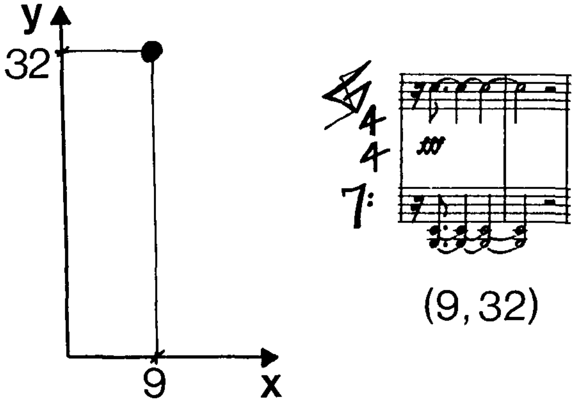


Fig. 3. The horizontal coordinate x represents the first triad's interval and the vertical coordinate y represents the second interval.

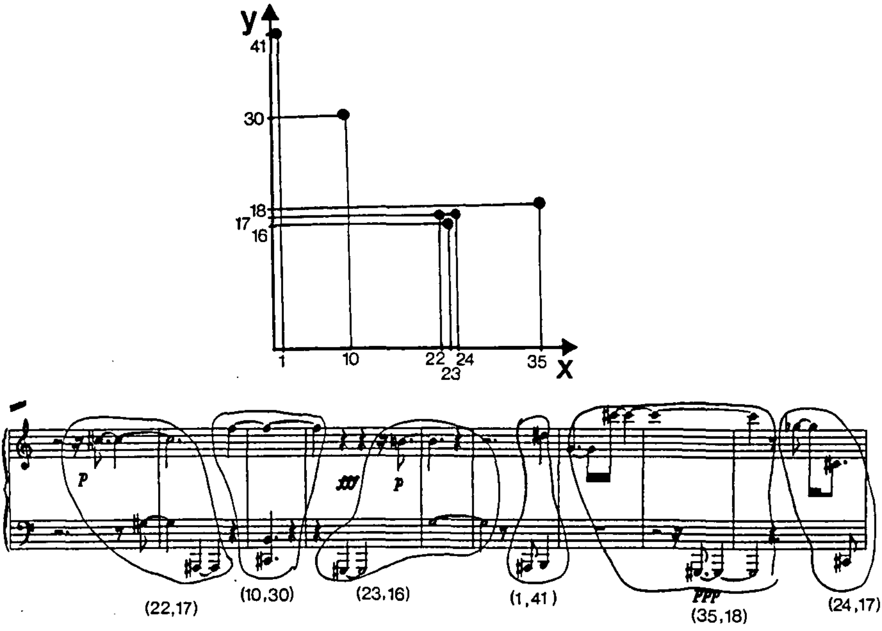


Fig. 4. The lattice of a musical passage represented by two-dimensional coordinates.

where *SG* is a set of control signals, sg_0 the ground state and *f* a local transition function (Good, 1968).

Consider

$SG = \{sg_0, sg_1\}$ such as $sg_0 = 0$ and $sg_1 = 1$.

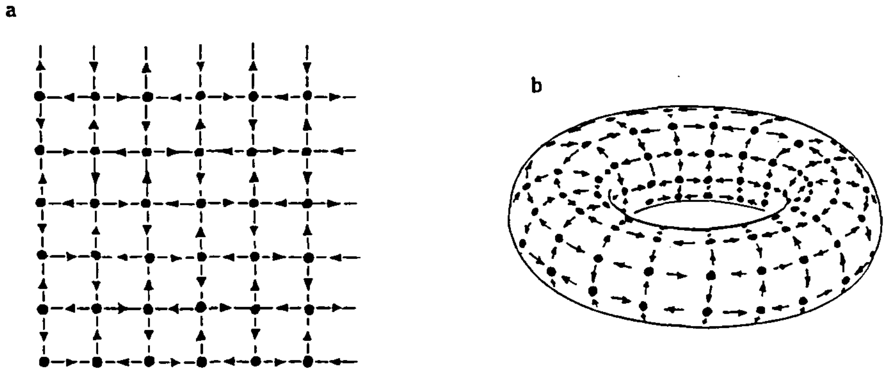


Fig. 5. The square lattice of cells (a) viewed as drawn onto a torus (b).

The above considerations represent a finite automaton which can assume two states: a quiescent (*0*) and a non-quiescent (*1*) state.

A configuration *c: X x Y → SG* defines a set of cells, where the elements are Cartesian coordinates of a two-dimensional interval space corresponding to non-quiescent state cells of *c:*

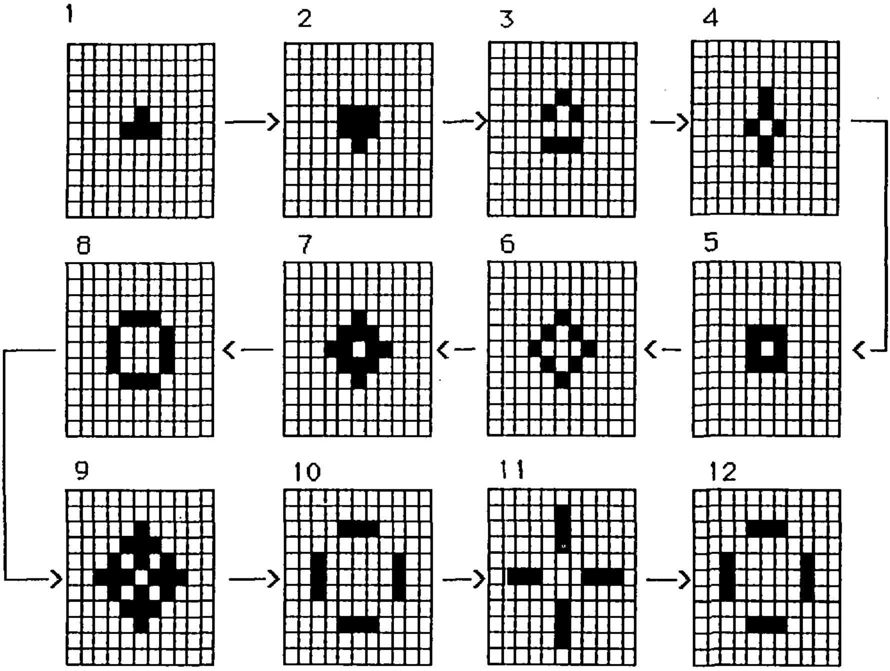


Fig. 6. Propagation of patterns arising from an initial configuration c_0.

$\{i \in X\, x\, Y\, /\, c\,(i)\ not = sg_0\}$ where i = interval series

A global transition function *F*, also called a ***transition rule***, drives the propagation $c_0, c_1, c_2, ..., c_t$. Each configuration *c* corresponds to a set of triads (Fig. 6).

In musical terms, the propagation of a configuration under a given global function can be viewed as a macro-formal organization of musical discourse in time. The microformal organization deals with the internal organization of each cell and between cells at time *t*.

The above concept is inspired by Xenakis' ***book of screens*** used as a graphical representation of sonic events in a slice of time (Xenakis, 1971).

CAMUS performs two different finite automata mapped into a *von Neumann* musical space. The first one is the *life* cellular automata algorithm which can assume two states: *0* (dead) and *1* (alive). It is used to work out a set of triads for each time step *t*, given the fundamental pitches. Also it is used to find out the aforementioned cellular microformal organization. The other is the *demon* cellular automata algorithm which can assume *n* states (*n* is determined by the user). It is responsible for the orchestration. Each state corresponds to an instrument to perform the triad. These two automata work in parallel and the music is performed in real time.

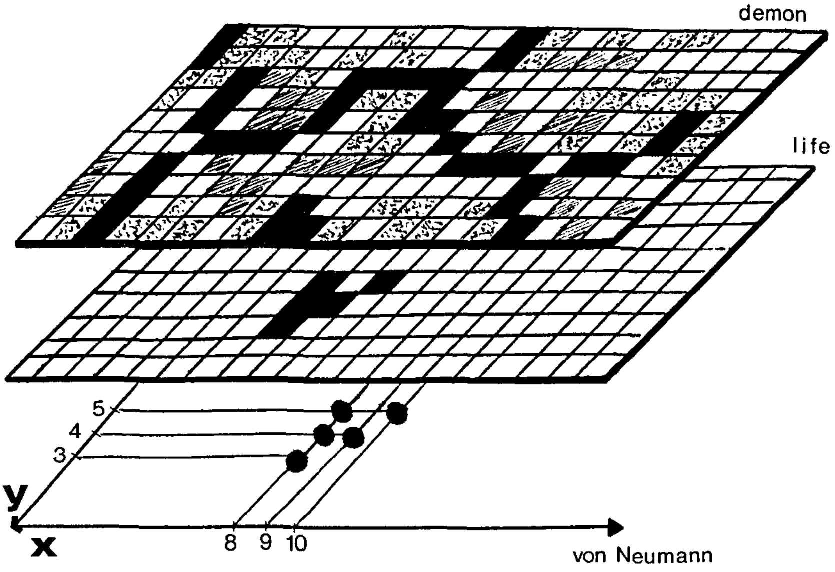


Fig. 7. CAMUS performs two different finite automata mapped into a von Neumann musical space.

Cellular genetic code and temporal reasoning

Each cell has its own duration but the pitches within a cell can assume different durations and be triggered at different times, forming different shapes. These shapes are determined by a predefined CAMUS codification: the so called 'cellul Ar geNetic coDe'.

The cellulAr geNetic coDe - AND

Like a DNA molecule in the cell's nucleous, whose symbols, drawn from a meagre alphabet of four different chemical bases (adenide, guanine, cytosine and thymine) form strands – coiled up into a helix –, the genetic information of a cell is formed by a string whose symbols have temporal meaning.

Like a nucleotide pair, the genetic information string has two components: *(Tgg → Dur)*. Each of these components can be one of the following codes according to the temporal shapes they stand for. There are 10 different temporal shapes (Fig. 8). The combination of two shapes forms the so called *cellular typology*.

Consider, as an example, the following typology: *dna → d[na]* (Fig. 9).

To understand how a cellular typology has a meaning in time, imagine the above example in a time domain representation (Fig. 10).

The first genetic information string's components, *Tgg*, stands for trigger shape, and the second, *Dur*, for duration shape. So, each pitch within a cell has its individual trigger time and duration time:

$$dna \rightarrow d[na] = \{Tgg(D), Tgg(N), Tgg(A),\\ Dur(D), Dur(N), Dur(A)\}.$$

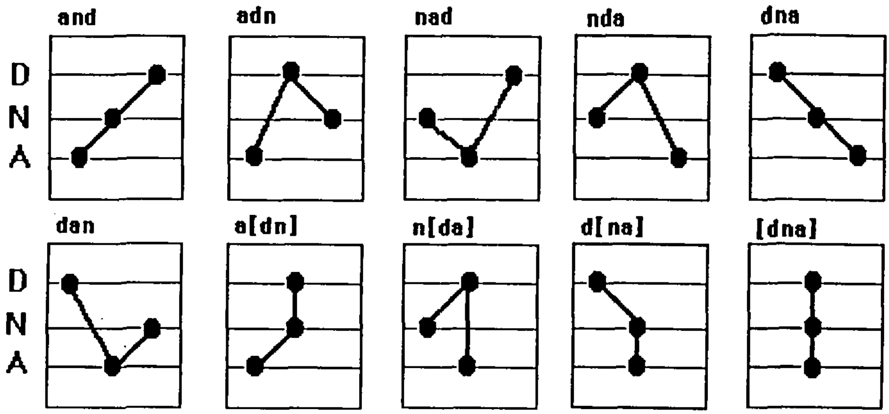


Fig. 8. There are 10 different temporal shapes. The combination of two forms a cellular typology.

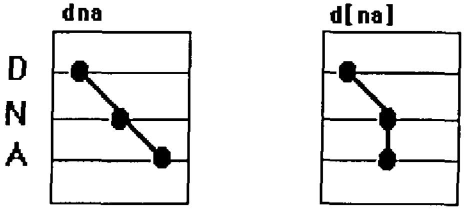


Fig. 9. Typology dna Æ d[na].

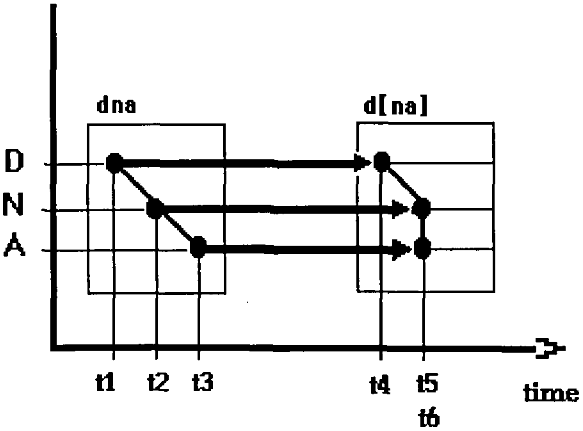


Fig. 10. Time domain representation for dna Æ d[na].

How to find out a cell's genetic information

AND is deduced by the life cellular automata. Considering that each cell to be played corresponds to a *von Neumann* space's non-quiescent cell and that each cell contains an identical copy of a two-state finite automaton we now propose a mechanism to find out the cellular genetic information.

Assume a 4 digit binary codification for the cellular genetic code as follows:

and	⇒	0101
adn	⇒	0010
nad	⇒	0111
nda	⇒	1011
dna	⇒	0011
dan	⇒	0110
a[dn]	⇒	0000
n[da]	⇒	1111
d[na]	⇒	1001
[dna]	⇒	0001

Then the Tgg and Dur string components are defined by:

$$Tgg ::= abcd \,/\, dcba$$
$$Dur ::= mnop \,/\, ponm$$

where

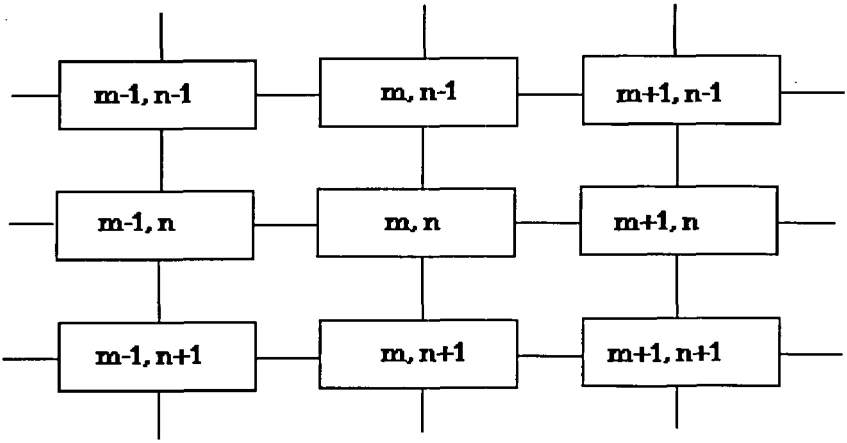
$$a, b, c, d, m, n, o, p \in \{0,1\}$$

The above codes are determined by the neighbouring cells (Fig. 11). Considering the geometrical relation below, the genetic information string is discovered as follows:

$$a = (m,n-1)$$
$$b = (m,n+1)$$
$$c = (m+1,n)$$
$$d = (m-1,n)$$
$$m = (m-1,n-1)$$
$$n = (m+1, n+1)$$
$$0 = (m+1,n-1)$$
$$p = (m-1,n+1)$$

Note that typology is just an abstraction of a shape. The numerical values for each individual pitch's trigger and duration are calculated when the cell is being performed, according to a chosen probability formula (Dodge, 1985). CAMUS calculates these values based on a distribution formula selected by the user.

By chance, there will be cases where one or more pitch's durations overflow the duration of a cell. This phenomenon is called overlap (Fig. 12).



m-1, n-1	m, n-1	m+1, n-1
m-1, n	m, n	m+1, n
m-1, n+1	m, n+1	m+1, n+1

Fig. 11. The neighbouring cells and their geometrical relationship.

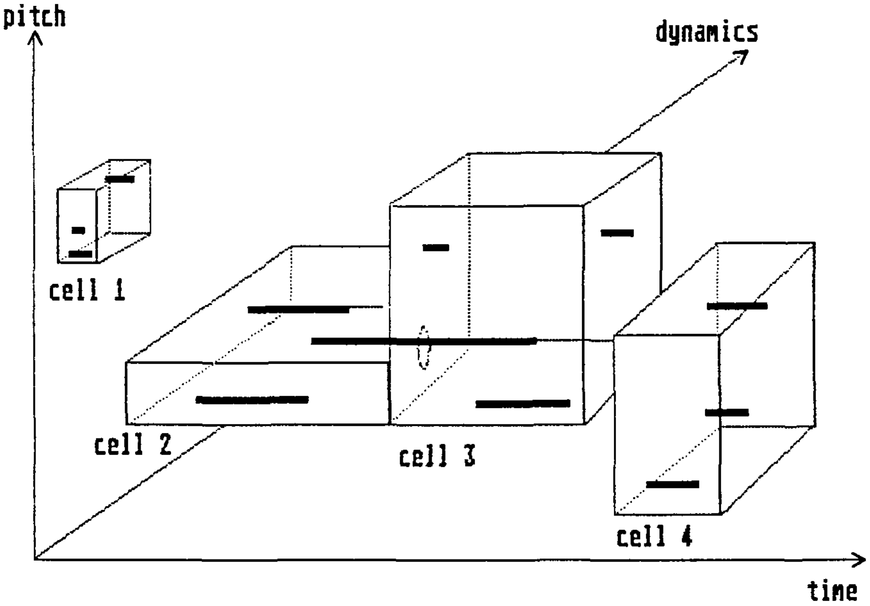


Fig. 12. Sometimes one or more pitch durations overflow the duration of a cell.

CAMUS AND MUSICAL THINKING

Who is the composer? The computer, the user, or the system's designer?

The problems of musical style and personal preference are very acute in the design of music systems for composition. Even the design of a general music language interface for music composition emerges from arbitrary processes, just as the goal – a composition – emerges from seemingly arbitrary processes. The relationship among the tasks comprising a compositional process for the manipulation of musical materials is not readily apparent (Pennycook, 1985).

Perhaps these questions could be altered to carry on this discussion. We suggest something like: "What sort of system are you using to make music?", "How open is this system for the musician's subjectivity", "What kind of thinking should the composer have to compose using a system like that?"

For a CAMUS thinking two important concepts should be taken into account: parametrical thinking and the articulation concept.

Since the beginning of the second half of this century composers have begun to think in terms of parameters. The term parameter refers to a variable control on some output. In the musical application, the output is some musical expression, and the parameter may refer to any musical feature which can vary between fixed limits. However, musical parameters have an infinite set of elements between fixed limits. A musical system may be viewed as a category of functions that defines subsets from those sets. In general, for computer music applications, they must be discrete and must be able to be described by recursion.

Basically CAMUS works on four dimensions of parametric control: time-pitch-timbre-dynamics.

By selecting an orchestration value, a set of different timbres can be specified to be active during the piece.

The user can specify different articulations to be performed through the piece. Articulation in this case is a group of configurations sharing certain common characteristics. Each articulation is specified in terms of:

i) *length* ⇒ how long it will be in terms of cellular automata time steps (in other words, number of configurations):

ii) *speed* ⇒ how fast its elements will evolve;

iii) *dynamics* ⇒ how loud its objects should sound;

iv) *fundamental pitches* ⇒ sequence of pitches to be used as the basis of cells.

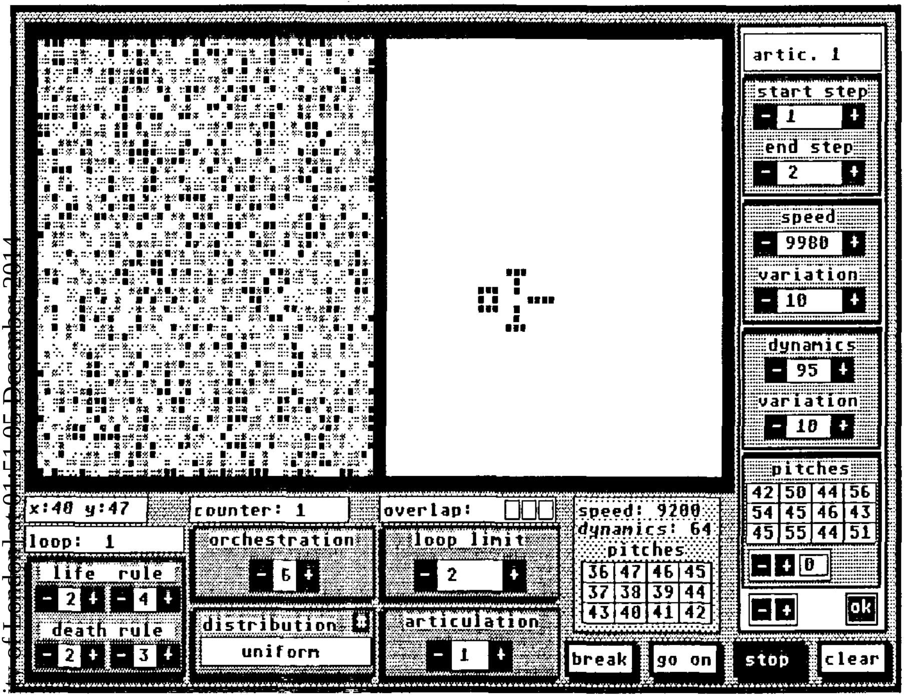
Musical form is also important and is related to the time domain. Form has to do with the placing of musical materials in time (Orton, 1990). It is possible to identify two levels of musical formal control in CAMUS. The lowest level has to do with micro-formal organization and the highest level with macro-formal organization. Micro-formal organization is entirely controlled by the cellular automata. However, it depends on the macro-formal organization specified by the composer as an initial setup: such as *life's* algorithm rule, where the patterns were drawn – as an initial state for life cellular automaton –, the orchestration, number of articulations, the articulations themselves, the probability formula – to initialize the *demon* automaton and calculate temporal values –, and so on. It is curious to observe how cellular automata preserve musical identities that operate within musical memory. The most important CAMUS' characteristic concerning musical form is that there is no exact repetition. I would say that cells are in a state of constant variation: there is a typology for each cell, a shape abstraction, but its values are always different. Composing with CAMUS is like the nature of an experimental action: *an action the outcome of which is not foreseen* (Cage, 1966).

EXPERIMENTAL IMPLEMENTATION

A CAMUS prototype (CAMUS V1.0) was implemented in order to compose some experimental music involving the above concepts (Fig. 13).

Essentially CAMUS V1.0 generates MIDI output from a parallel evolution of two different cellular automaton algorithms: Conway's *life game* and Griffeath's *crystalline spirals generator (demon)* (Miranda, 1991) (Fig. 14).

The life game cellular automaton is responsible for pitch selection. Each cell corresponds to a triad (tuple of 3 pitches): the first is given by a predefined sequence of MIDI pitch numbers defined by the user and the remaining two are



artic. 1

start step: 1

end step: 2

speed: 9980

variation: 10

dynamics: 95

variation: 10

pitches

42	50	44	56
54	45	46	43
45	55	44	51

0

x:40 y:47

counter: 1

overlap: □□□

speed: 9200

dynamics: 64

pitches

36	47	46	45
37	38	39	44
43	40	41	42

loop: 1

life rule: 2 4

death rule: 2 3

orchestration: 6

distribution: uniform

loop limit: 2

articulation: 1

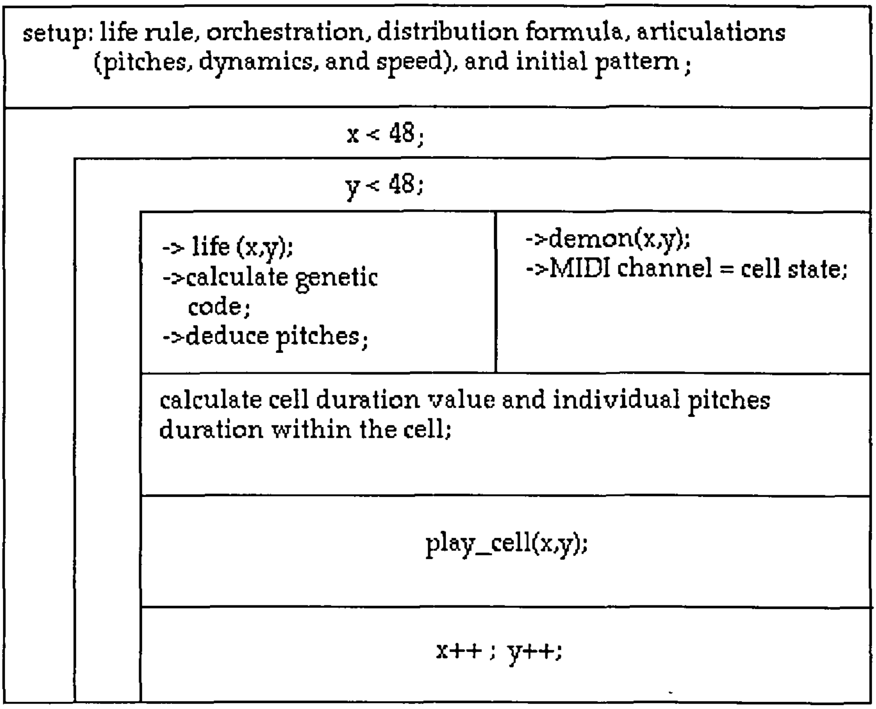
Fig. 13. CAMUS V1.0 main screen dump.

determined by the corresponding cells' coordinates in the von Neumann musical space (each coordinate corresponds to a pitch distance). The default life game algorithm is the one discovered by Conway, but other variants may be experimented with. Also the user "draws" the initial pattern

Each cell has its own duration according to a distribution formula and speed, both specified by the user. Also each cell has an internal typology. Typology is the shape which pitches can assume according to their individual trigger time and durations within a cell. The life cellular automaton is also responsible for typology selection according to certain genetic codes associated with its neighbouring states.

The demon cellular automaton is responsible for the orchestration. It determines which MIDI channel will be used to output a cell according to its state. The orchestration value is given by the user. Each channel should be able to play 6 polyphonic voices (the triad plus 3 possible overlaps from previous cells).

Different articulations may be specified to take place within a certain step's loop. Each cell in a cellular space changes its state in synchrony with a tick of an imaginary clock. Each tick (time step or simply "step") forms a configuration of cells to be performed. After specifying how many steps will be performed within



```
setup: life rule, orchestration, distribution formula, articulations
        (pitches, dynamics, and speed), and initial pattern;
x < 48;
    y < 48;
        -> life (x,y);                 ->demon(x,y);
        ->calculate genetic            ->MIDI channel = cell state;
           code;
        ->deduce pitches;
        calculate cell duration value and individual pitches
        duration within the cell;
        play_cell(x,y);
        x++ ; y++;
```

Fig. 14. The logics of the program in overall form.

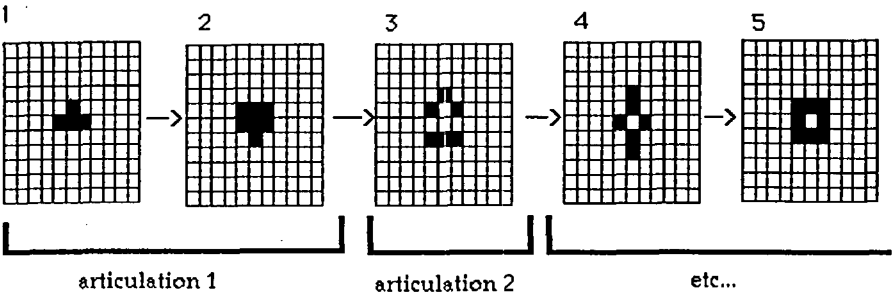


Fig. 15. Different articulations may be performed within a loop. Each one has its own speed, dynamics and fundamental pitches.

a loop, the user can specify how many articulations will take place there. Each articulation has its own speed, dynamics and fundamental pitch sequence (Fig. 15).

VORTICAL MOTION: A CELLULAR AUTOMATA MUSIC CONCEPT FROM FLUID MECHANICS

Perhaps vortical motion is not a suitable concept to be considered for this version of CAMUS, but for further implementations that will deal with sound synthesis and spatial localization of sounds.

This concept, well known from fluid mechanics and widely applied in electroacoustic music (Lewis, 1988), has to do with the types of motion which have ostensibly vortical qualities. There are three qualities of vortical motion to be identified:

i) *Cyclicity:* Vortices are associated with rotational motion, usually rotation at a constantly accelerating and decelerating rate. Rotating objects create repeated patterns when observed from fixed points. This makes rotational motion particularly suited to cellular automata music, since cyclicity has historically been very important in music and cellular automata basically work upon cycles of patterns;

ii) *Fluidity:* Fluidity is not associated with the movement of solid objects but rather fluids. The movement of water, for example, is bound up with the idea of cyclicity and repetition. The use of continuously changing parametric values for sound synthesis results in clouds of sounds that suggest the amorphous and unstable nature of fluids. The implementation of granular synthesis (Roads, 1991) controlled by means of a cellular automaton will enable composers to think in terms of fluidity of musical events when using CAMUS;

iii) *Spatial motion:* This effect deals with the psychological perception of spatially moving sound structures. For example, an extended glissando would suggest imaginary objects ascending or descending. A cell's typology can be extended to spatial localization for specification of a sort of spatial morphology for cells, determined by their neighbourhood.

FURTHER DEVELOPMENTS AND CONCLUSION

This paper described an attempt to work out a system for composition inspired by cellular automata. Any systematization of a technique to compose music leads to limitations. It is up to composers' preferences and aims whether to use them or not.

This system is designed for a very abstract mode of composing. It needs aural imagining to foresee the results. But it is very limited to a certain world of musical possibilities. However, it is a very good tool for experimentation. Personally I have been using CAMUS to compose musical events to be used in certain passages of major works.

I would regard this work as a starting point and as an invitation for further investigations in this field.

Perhaps the most plausible further step is to extend this technique to sound synthesis. Many attempts have been made to model granular synthesis by means of cellular automata. Granular synthesis involves generating thousands of very short sonic particles to form larger acoustic events (Roads, 1991). Each particle would be associated with a single automaton – a cell. The generation of musical events would be driven by a cellular automata algorithm.

ACKNOWLEDGEMENTS

I would like to express my gratitude to R. Orton, R. Kirk and A. Hunt of York University for their support at the early stages of this work. I am also grateful to P. Nelson of Edinburgh University who read a draft of this paper and made valuable suggestions.

REFERENCES

Cage, J. (1966). *Silence* . London: Marion Boyards Ltd.

Cood, E.F. (1968). *Cellular Automata.* New York/London: Academic Press.

Dewdney, A.K. (1985). Building computers in one dimension sheds light on irreducibly complicated phenomena. *Scientific American,* May, 10-16.

Dewdney, A.K. (1989). A cellular universe of debris, droplets, defects, and demons. *Scientific American,* August, 88-91.

Dodge, C., & Jerse, J.A. (1985). *Computer Music: synthesis, composition and performance.* London/New York: Shirmer Books.

Forte, A. (1973). *The Structure of Atonal Music.* Harvard: Yale University Press.

Höller, Y. (1984). Composition of the Gestalt, or the making of an organism. *Contemporary Music Review,* Tod Machover (ed), *1*:1, 35-40.

Lewis, A. (1988). Motion and the analysis of electro-acoustic music: Denis Smalley's 'Vortex'. *Electro-Acoustic Music,* Electro-acoustic Music Association of Great Britain, *3*:3, 16-22.

Miranda, E.R. (1991). Cellular Automata Music investigation. *Music Technology MSc dissertation,* University of York, York.

Ortőn, R. (1990). Music Fundamentals. *Music Technology lecture notes,* University of York, York.

Pennycook, B.W. (1985). Computer Music Interfaces: A survey. *Computing Surveys,* Curtis Abbott (ed), 17:2, 267-289.

Roads, C. (1991). Asynchronous Granular Synthesis. *Representation of Musical Signals,* Giovanni De Poli et al. (eds), The MIT Press, 143-186.

Xenakis, I. (1971). *Formalized Music.* Indiana University Press.

APPENDIX

THE VON NEUMANN MUSICAL SPACE (vNMS)

Let **P** denote a set of integers and **a** a cell. To obtain a vNMS we associate with the set **P x P**:

a) the *neighbourhood function* **g:P x P** → **2** $^{\mathbf{P \times P}}$ defined by **g(a)={a+b$_1$, a+b$_2$ +..., a+b$_n$)** for all **a** ∈ **P x P** where b$_i$ (i=1, 2, 3, ..., n) ∈ P x P is fixed.

b) the *automaton* **(M, m$_0$, f)** where **M** is a discrete set of finite states, **m$_0$** is a distinguished element of **M** called *quiescent state,* and **f** is the local transition function from n-tuples of elements of **M** into **M, f:M** → **M,** subject to the restriction **f (m$_0$ m$_0$, ..., m$_0$) = m$_0$.**

Note that such a set **M** of states must be essentially isomorphic to a finite set of real numbers, such as: **m$_o$=0, m$_1$=1, m$_2$=2, ..., m$_n$=n.**

It is convenient to think of the vNMS as a plane assemblage of a countable number of interconnected cells. The location of each cell is specified by its cartesian coordinates with respect to some arbitrary chosen origin. Each cell contains an identical copy of the automaton **(M, m$_0$ f),** and the state **m^t(a)** of a cell **a** at time **t** is precisely the state of its associated automaton at time **t.**

Each cell **a** is connected to the **n** neighbouring cells **a+b$_1$, a+b$_2$ ..., a+b$_n$.** In all that follows we shall assume that one of the neighbours of **a** is **a** itself, and in accordance with this assumption, we adopt the convention that **b$_1$=0.**

The *neighbourhood state function* **h^t:P x P** → **m^t** is defined by **h^t(a)=(m^t(a), m^t(a+b$_2$), m^t(a+b$_3$), ..., m^t(a+b$_n$)).**

Now we can relate the neighbourhood state of a cell **a** at time **t** to the state of that cell at time **t+1** by **f(h^t(a))=m^{t+1}(a).**

By restricting our attention to the case in which all cells except a finite number are initially quiescent, on **f, f (m$_0$ ..., m$_0$),** means that a nerve cell whose neighbourhood is entirely quiescent remains quiescent itself. Thus at every time step all cells except a finite number are quiescent.

An allowable assignment of states to all nerve cells of the vNMS is called a *configuration* (or the *vNMS state).* Thus, a configuration is a function **C: P x P** → **M** such that **{a** ∈ **P x P | c(a) not = m$_0$}** is finite.

Such a function is said to have *finite support* (or *finite configuration assumption*) relative to **m$_0$,** and the above set is denoted **sup(c).** Thus, a configuration **c** is an assignment of a vNMs state **c(a)** to each cell **a** in **P x P** such that only a finite number of cells is assigned states other than **m$_0$.**

We are now in a position to define the *global transition function* **F.** Let **C** be the class of all configurations for a given vNMS. Then **F: C x C** is defined in terms of the local transition function as **F(c) (a)=f(h(a))** for all **a** ∈ **P x P.** Given an initial configuration **c,** the function **F** determines a sequence **c^0, c^1, c^2, ... c^n,** called *propagation,* where **c^{t+1} = F(c^t).**

The study of a propagation must involve the interpretation of the sequence **c, F(c), F^2(c), ..., F^t(c), ...**

We wish now to distinguish between configurations which change with time and those which do not. A subconfiguration **c** is called *passive* if **F(c)=c.**

The following definitions attempt to make precise whether configurations interact with one another. The configurations **c** and **c′** are *disjoint* if **sup(c)** ∩ **sup(c′).** If **c** and **c′** are disjoint, their union is defined by:

i) (c U c′) (a) = c(a) if a ∈ **sup(c);**

ii) (c U c′((a) = c′ (a) if a ∈ **sup(c);**

iii) (c U c′) (a) = $\mathbf{m_0}$ otherwise.

Similarly, if **c** is a configuration of **d**, the configuration **c-d** means the configuration **e** such that **c U e = d.**

not (sup(c)) means the complement with respect to the entire space of **sup(c).**

In order to talk about *composition* within the model, we must be able to interpret a sequence of states of the model in such a way as to say that a particular configuration has or has not been composed after a certain number of time steps. It seems quite reasonable to require that we interpret as composed only those configurations appearing in areas which at time **t=0** were entirely quiescent. Thus the following definition is proposed:

i) A configuration **c** composes a configuration **c′** if there exists an area **A,** disjoint from **c,** and a time **t** such that $\mathbf{c'=F^t(c) \mid A}$.

Example:

Consider the model with **M={1,0}**, $\mathbf{m_0=0}$ and for all $\mathbf{m_i}$,

$$f\,(m_1, m_2, m_3, m_4, 1)=1$$
$$f\,(m_1, m_2, m_3, m_4, 0)=m_1,$$

where **F(c) (a)=f(c(a)), c(u(a)), c(d(a)), c(r(a)), c(1(a))),** considering the following 4 neighbours geometric relation:

$$u(x,y) = <x+1,y>$$
$$r(x,y) = <x,y+1>$$
$$d(x,y) = <x-1,y>$$
$$l(x,y) = <x,y-1>.$$

At time **t+1** a cell **a** remains in its state of time **t** if its neighbour to the left is in state **0.** On the other hand, if the state of the left-hand neighbour is **1** at time **t,** then the state of **a** is **1** at time **t+1.**

Eduardo Reck Miranda
Faculty of Music
University of Edinburgh
Alison House
12 Nicolson Square
Edinburgh - EH8 9DF
Scotland - UK
MIRANDA@UK.AC.ED.MUSIC

Eduardo Reck Miranda, born in Porto Alegre, Brazil, on October 20, 1963. Studied electroacoustic music in Sao Paulo, Rio de Janeiro, and Brasilia, between 1986 and 1989. Completed a degree in Data Processing Technology (UNISINOS, Brazil) in 1985, and MSc in Music Technology (University of York, England, UK) in 1991. Meanwhile studied Philosophy, Discrete Mathematics, and Music at Federal University of Rio Grande do Sul, Brazil, where he also has worked as a composer and researcher in Computer Science applied to Music.

Nowadays he is reading a joint Ph.D. at the Music Faculty and Dept. of Artificial Intelligence of Edinburgh University, in Scotland, UK. His research work involves Artificial Intelligence aided Signal Processing and Algorithm Composition.