Cellular automata with special reference to their implementation using associative memories

A literature review

A G Turner

Department of Computer Science,

University of York

March 13, 1997

Abstract

This literature review discusses the implementation of cellular automata using associative memories as the state machine. Some historical context is described, as are associative memories and brief notes on hardware implementations

1 Introduction

Cellular Automata (CA) are systems of small interconnected elements, or cells. Typically they are represented as one dimensional or two dimensional grids although in theory any geometric shape is possible [66]. The cellular automata progresses through a series of states, or configurations, through successive iterations, and it is this progression that is of interest. Each of the cells in the automaton has a state which is chosen from a set of possible states. Typically the cells in an automaton are identical apart from the possibility of each having a different state. For example, in Conway's Life, a familiar example, any particular cell may either be "alive" ("on") or "dead" ("off"), there being only two states.

The biological analogy is often used - for example comparing a cellular automata to a colony of bacteria, living, competing, dying and giving birth, which is the analogy Conway used when developing Life. In such biological systems the behaviour of a cell is often dependent on the behaviour of cells nearest it - i.e. each cell only reacts to its local conditions, but the behaviour of a colony of bacteria can be seen in a petri dish. In some ways the biological analogy is weak because in many natural systems there are complex rules in play, but study of cellular automata has shown that even with fairly basic local interactions complex behaviour can be seen on the macroscopic level.

Typically all the cells in the automaton are identical, since this makes calculations on a computer running the

simulation easier, and also makes mathematical analysis of the system far simpler. This need not be the case and interesting results have been obtained by having cells of different types, which is more akin to a typical multicelled animal, for example.

2 Definitions

The following assumes that all the cells in the CA are identical in form, i.e. the CA is homogeneous. The notation is derived from Codd [4]

The whole automaton space, typically two dimensional is represented as a set of cells IxI.

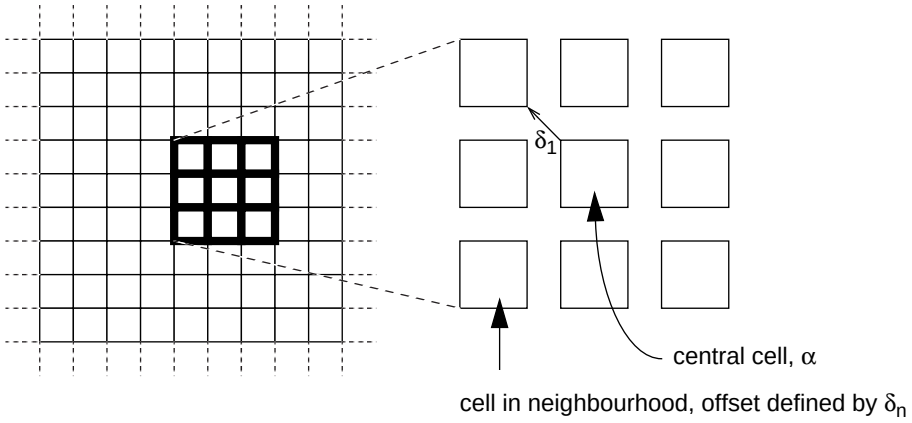
2.1 States

Each cell, a, has a state chosen from the set of all possible states. The state of a cell at a particular time, *t*, is represented by $v^t(a)$. One of the states is given special significance and is termed the quiescent state, v_0. This state is generally considered to be a resting, null, off or empty state, depending on the particular interpretation placed on the automaton. For example, if the automaton were being used to model a quantum gas the quiescent state would be interpreted as the absence of a particle.

2.2 Neighbourhood

A cell also has connections to other cells, which are termed the neighbourhood of the cell. [2][3][4][5][16][21].

figure 1 - a neighbourhood in a 2d dimensional rectangular CA



part of a cellular automaton **a cell and its neighbourhood (or neighbourhood template)**

Cellular automata depend on the fact that the direct interactions for a particular cell are just with the elements of its neighbourhood, and no further afield.

The neighbourhood is defined by a template, which is the pattern of the neighbourhood, by the function:

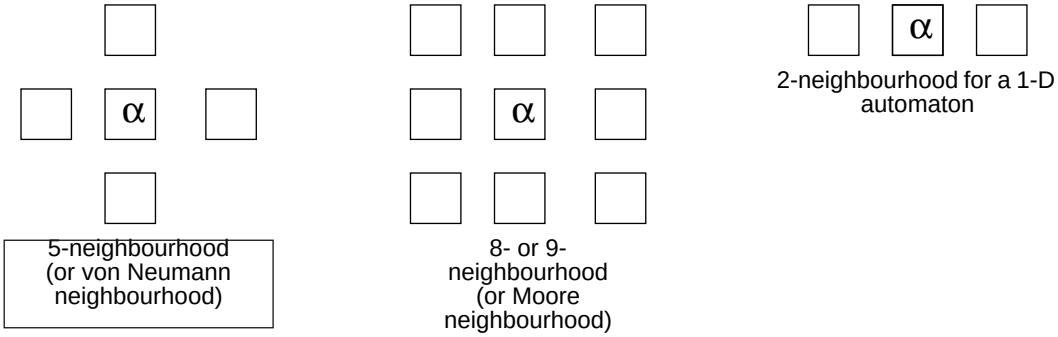
$$g(\alpha) = \{\alpha + \delta_1, \alpha + \delta_2, \ldots \alpha + \delta_n\} \quad \text{for all } \alpha \in IxI \qquad \textbf{equation 1}$$

where δ_n is an offset from the central cell to a cell in the neighbourhood, $\alpha+\delta_n$

This indicates that *g* maps from α to the members of the neighbourhood, via a number of offsets. There is a certain confusion in terminology: some terminologies include the central cell, α, as part of the neighbourhood, whereas others exclude it from the neighbourhood but consider it to be included in the kernel, which is a superset of the neighbourhood also including the central cell.

Typically neighbourhoods are referred to by the number of elements in the neighbourhood or kernel, or by the names of those that first made great use of them.

figure 2 - some examples of neighbourhoods



2.3 Local Transitions

The new state is derived from a function based on the states of the cells in the neighbourhood of the cell in question *and* the state of the cell itself. It may be that the function is invariant for some elements of the neighbourhood. e.g. the central cell.

If we define the state of a cell α at a time *t* to be $v^t(\alpha)$ then the new state of the cell is

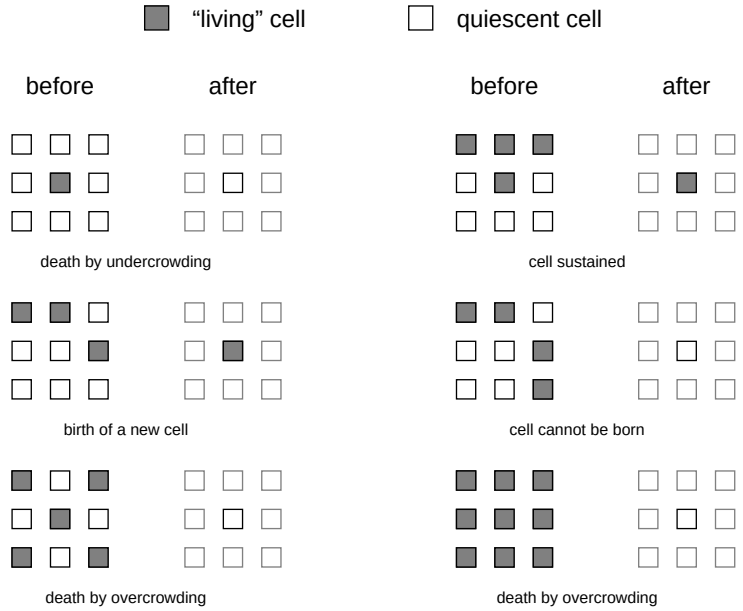
$$v^{t+1}(\alpha) = f(v^t(\alpha), v^t(\alpha+\delta_2), \ldots, v^t(\alpha+\delta_n)) \qquad \textbf{equation 2}$$

which states that the new state of the cell at time $t+1$ is dependent on the states of the cells in the neighbourhood at the previous instant, *t*, via the local transition function, *f*.

The local interactions means that cellular automata may be considered to be highly parallel which has important implications for their implementation.[9][13][19].

A local transition function is actually composed of a number of simple rules that indicate what the next state of the cell will be given the current state of the cell and cells in the neighbourhood. Examples from Conway's Life are shown in figure 3

figure 3 example selected local transition rules from Conway's Life



2.4 Global Transitions

The local transition functions may be clear in local effect but the global behaviour is less clear as it emerges from the local behaviour. The new state of the whole automaton is determined by the global transition function:

$$F(c)(\alpha) = f(h(\alpha)) \quad \text{for all } \alpha \in IxI \qquad \textbf{equation 3}$$

h is derived from the states of the neighbourhood, i.e.

$$h^t(\alpha) = (v^t(\alpha), \ldots, v^t(\alpha + \delta_n)) \qquad \textbf{equation 4}$$

which gives us a progression of configurations via

$$c_{t+1} = F(c_t) \qquad \textbf{equation 5}$$

This indicates that the new state of the whole automaton which is the *configuration*, c, is created by the action of the local transition function on each cell and its neighbourhood with the states as at time t. i.e. we record the old states of the automaton, calculate the new states based on the states at time t and transfer the new results to a new copy of the automaton.

2.5 Configurations and Subconfigurations

A configuration is a particular arrangement of states of cells in an automaton such that only a finite number are in the non-quiescent state. As can be seen from equation 4 a configuration passes from one configuration to another under the action of the global transition function, but the new configuration will always be finite. While the whole set of states in the automaton is a configuration there may be groups of non-quiescent cells surrounded by cells in the quiescent state that never come into contact with other groups of cells. These knots of cells are termed subconfigurations, being disjoint in that they receive no information from other subconfigurations [4][9][10] (see figure 5).

figure 4 - a configuration mapping to another under the global transition, F

this example again uses Conway's Life

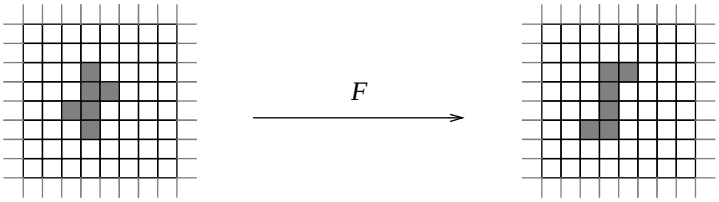
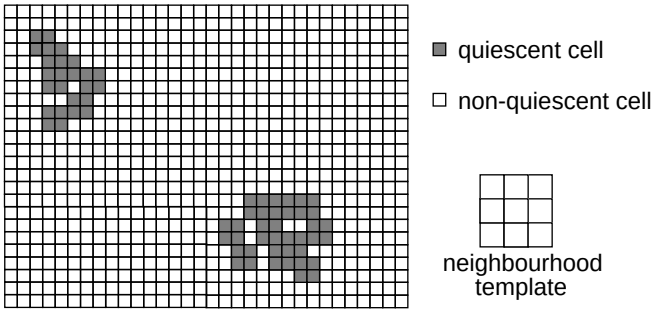


figure 5 - two disjoint subconfigurations



3 Studying Cellular Automata

The study of cellular automata is, essentially, the study of the way that one configuration progresses into another, given a certain local transition function. The global transition function is, in general, impossible to determine from the local transition function and cellular automata are highly complex systems. Studying them is worthwhile, however, since the microscopic rules of a system may be known even if the behaviour of a particular instance of the system over time is not.

For example, the laws of a gas, at a classical level, are those of billiard balls bouncing off each other, and these can be easily modelled with a CA (give an example). Having set up this, an automaton can be populated with cells in such a way that some have states that represent gas particles travelling in a particular direction, and the behaviour of the "gas" in that instance can be observed.

3.1 Complexity and Classification

In general the global transition function is, in general, hard to predict given even simple local transition functions and cellular automata are generally highly complex systems with emergent behaviour [28][59] which may seem chaotic. Wolfram and Langton spent some time attempting to classify various cellular automata according the complexity of the global behaviour [51].

Wolfram classified CA into the following classes:

- Automata that always end up with a certain configuration no matter what the starting configuration.
- Automata that get stuck in cycles of states.
- Automata with long lived localized patterns and local order.
- Chaotic automata that never settle down.

Ultimately the only way to be absolutely sure how a CA will perform is to run the system [60].

3.2 Reversibility

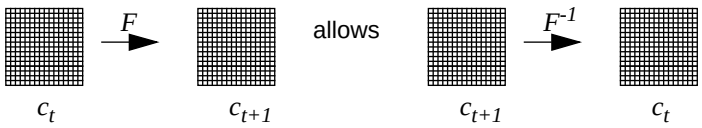
As noted above, some cellular automata converge to a particular configuration no matter what the starting configuration, and these automata are termed *irreversible*. For a cellular automata to be reversible [65] it must be possible to create an "inverse" cellular automaton.

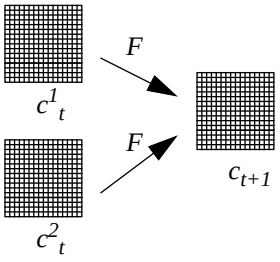
i.e. for an automaton A for which $c_{t+1} = F(c_t)$ (equation 5)

there must be and automaton A^{-1} for which we have $c_t = F^{-1}(c_{t+1})$

This can only be the case there is no other configuration for which A also produces the configuration c_{t+1}.

figure 6





means no single valued function F^{-1}

Moore noted the existence of "Garden of Eden" configurations [6] that could be created by at time $t=0$ yet could not be created by finding a reverse transition function F^{-1} and applying it to the configuration at time $t=1$. Hence if

a "Garden of Eden" configuration exists then the automaton is not reversible. Myhill [76] examined the converse. Toffoli also looked at the question of reversibility since reversibility suggests various thermodynamic properties concerning the entyropy of cellular automata and allows well known techniques from physics to be employed (Toffoli and Wolfram were originally physicsits). Toffoli asserts that reversible cellular automata are easy to find [56] although this is a controversial suggestion, countered by Culik [70]. Reversibility may have implications concerning the ability of an automaton to back track after a component failure.

3.3 Emergent behaviour

Emergent behaviour [28][59] may also provide great strength should the components be unpredictable, this idea being initially expounded in the paper by John von Neumann [29], backed up by more recent research by those such as Gacs [67][68]. The fact that reliable systems may in theory be constructed from unreliable elements means that cheap and nasty implementations of the state machines (i.e. cells) may be good enough. However, only some cellular automata can recover from such errors, and for some applications where a exact answer rather than some general behaviour is required it can be disastrous.

For an automaton to process information and initial configuration of cell states must first be set up [4]. Processing occurs when new states for each of the cells is calculated. The new state of the cell is determined by a function of its current state and the state of cells in the neighbourhood. This function is termed the local transition function [4][10][11] and maybe deterministic or contain non-determinstic elements [9][19]. The cells of the automata have one special state, the quiescent state, which is considered to be a null or off state.

3.4 Inhomogeneity

The classic cellular automata assumes that all the cells are indentical, but this is rather limiting when compared to natural systems which often have a variety of elements. Von Neumann devised a complex 29-state system that was effectively an attempt to model an inhmogenoeus system while retaining just a single local transition function. More recently inhomogeneity has been examined [71][73], although it is less readily addressed by mathematics due to the additional complexity.

4 Use of Cellular Automata

There are two main uses of automata

- computation where the local transition function models some process, e.g. a physical process such as heat flow, or a lattice gas [2][3][16][14][15][20][54][55][56][60][61].
- A more general computing machine is built inside the cellular automata by choosing special configurations of states to emulate parts of a computer and represent data. Thus the cellular automata effectively becomes a special form of parallel computer [8][62][63][64].

4.1 Turing Machines and Self-Reproduction

The term self-reproduction is difficult to define [58] and is strictly defined in terms of Turing universal construction [12][10][11][5][7][8][21][22][23]. Universal construction is rather a strict definition and if applied to humans they would not be considered self-reproducting. Langton prefers the term 'active self-reproduction'. self-repdroduction is essentially the ability to create a copy of the current subconfiguration as (ultimately, when construction is ended) a disjoint subconfiguration, but for it to be an active construction some uniterpreted data must also be passed to the newly constructed sub-configuration such that it has data to operate, and in the context of an automaton this is subsumed in the subconfiguration.

4.2 von Neumann and simplifications

John von Neumann first deduced a cellular system that would be able to reproduced a subconfiguration, but his system was unwieldy in the extreme. Describe briefly.

It was simplified from 29 states and eight nearest neighbours to just eight states (but more nearest neighbours) by Codd [4] and Moore [6] and further refined by Langton [58] while still retaining the self-reproduction ability. While these systems allowed self reproduction little was demonstrated apart from this ability - i.e. little was shown of an ability to do useful computation even if theoretically possible.

Meanwhile Conway simplified the system further producing his version Life [53] that was a binary automaton. This excited great interest since a binary system was very suitable for computer implementation. It had the disadvantage that it would require 10^12 cells to allow reproduction.

The binary system was taken up, ignoring the self reproduction aspect, for work on simulated annealing [16][17][18] and simplified ultimately by Wolfram [51] to a one-dimensional system that still gave insights into physical systems of diffusion.

4.3 Simulated Annealing

5 Hardware Implementation

The parallel nature of automata was noted and attempts to build hardware implementations exploiting this were attempted. Notable examples are

- ILLIAC
- CAM-6 [19][60] which also has a supporting language.
- CELIP (also with supporting language). [24]
- The Connection Machine (with supporting language). [62]
- CAM-BRAIN [63][64].

Ideally one would reserve one processor for each cell but this can lead to fairly inflexible systems and other parallelisation processes are now in vogue. Also one cell per processor relies on fast communication even when

only local information need be passed.

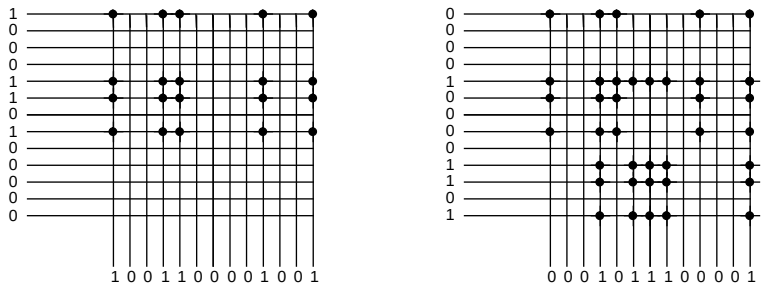
The classic implementation became the CAM machine of Toffoli and Margolus [19][60] which was flexible and was supported by its own powerful if idiosyncratic computer language based on Forth. When introduced it cost $1500, plugged into the back of a PC and ran CA more quickly than a Cray. Other languages have been developed to support cellular automata, unfortunately they all tend to be rather idiosyncratic, e.g. CELIP [24]. Predicate logic is sometimes used as an interface for building up local transition functions.

Using parallel hardware introduces an interesting question - synchronous or aysnchronous? Should the system wait until all cells have finished processing an iteration or just charge on regardless? This will depend on the system's ability to deal with unreliable results. The transition functions above presume that the cells of the automaton are updated synchronously, but this need not strictly be the case (although freeing this restriction makes mathematical analysis far more difficult) and asynchronous CA are possible, which is more closely analogous to, say, cells in an organism [28][52]

6 Associative Memories

These are systems which store associations between input vectors and output vectors allowing latter recall of the output vector given part of all of the original input vector [46][47][48][49]

figure 7 - a binary correlation matrix memory



teaching first association teaching a second association

In general part of the input vector may be enough, i.e. partial matching is supported The input vectors may be continuous but implementations using binary vectors are simpler [30][31][46].

6.1 Correlation Matrix Memories

The classic implementation of this is the correlation matrix memory (CMM)[46][47][31][30][36][34]. CMMs allow associations to be stored by employing a binary weights matrix. When a bit set to 1 in the input vector matches a bit set to 1 in the output vector then the corresponding point the weights matrix is set to 1, else left at its current value. Such binary weights allow compact storage as some information is allowed to be lost in the hope that the recovery process can reconstitute the lost information given some additional information concerning the general form of the input and output vectors

To ease recovery the input all output vectors during teaching should have the same number of bits set so a thresholding can be done on the results of the initial application of a test vector, the number of bits expected to be set being known. [30][31][34][36][46][48]

6.2 ADAM

The ADAM system of Dr. J. Austin [30][31][35][36] is an enhancement using two linked correlation matrices and an additional vector. This system allows more flexibility in the input and output vectors and allows numbers of input vectors to be more compactly associated with an output vector with less risk of saturation.

7 Implementation of Associative Memories in Hardware

Implementation of associative memories, specifically ADAM, in hardware had been investigated by SAT I, II and soon III will implement ADAM at York, with an AURA system to implement correlation matrix memories alone in the pipeline.[32][38][39][42][43][45][41]

8 Why Implement a Cellular Automaton Using Associative Memories?

There are four main advantages to implementing cellular automata using associative memories

- A state machine that is flexible enough to cope with a wide variety of rules with appropriate coding of states
- reasonably fast for complex local transition function through efficient storage of associations
- statistical parallelism may be used
- The ability to use specialised hardware already in existence for CMMs without having to build hardware specialised for cellular automata

Many hardware implementations of cellular automata have previously been fairly specific and detailed and not able to cope with additional complexities. However, Associative memories would only be a replacement for the state machines and additional control software would be required.

The disadvantages are

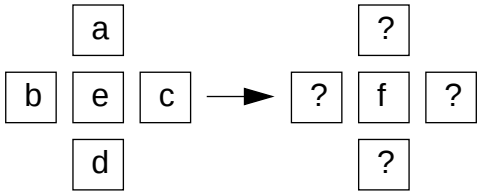
- that associative memories are not guaranteed to be fault free and so some applications may be in danger of not giving the desired result unless they are optimised to run on unreliable components, which implies a certain degree of self-organisation.
- training sets may be excessively large unless symmetries are identified and exploited

8.1 Training

To teach an associative memory to perform as the state machine in a fairly naive way we set up vectors to represent each state and concatenate them

figure 8 - a simple training scheme

given vectors represented by a, b, c, d, e, and f with the current state of the cell being e and the next being f then



would be implemented by the association between (abcde) and f

Thus we might represent a with 1100, b with 0110, c with 0011, d with 1001, e with 1010, f with 0101

Then we teach the association 11000110001110011010 with 0101

The set of possible inputs, given this scheme, may not be sufficiently orthogonal [46] as is required for efficient associations and some form of input preprocessing using tupling would be required. Given the statistical nature of teaching and recall with associative memories there is a small chance of errors occurring but the system can be fine tuned to make the possibility of error small and allow the errors to have minimal effect on any final outcome. [29]

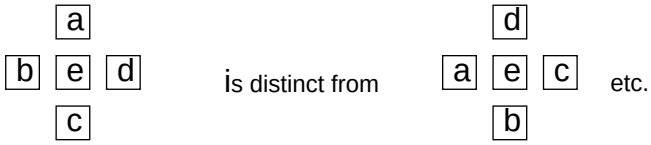
8.2 Statistical Parallelism

Using associative memories also allows us to employ the emerging techniques of statistical parallelism [36][44] to have multiple cellular automata running on the same set of associative memories offering large potential speed ups. Essentially SP allows various associations for a number of strands of computation to be stored simultaneously in a single correlation matrix memory. Input vectors are tagged with patterns indicating the computation strand to which they belong and an output vector containing the results of all associations is output that may be post-processed to reveal the result of each individual association. The advantage for implementing cellular automata is that a number of automata may operate simultaneously at high speed using a single set of correlation matrices or underlying hardware associative memory processors.

8.3 Coding symmetries in rules

There are problems, however, because of the basic teaching schemes inability to generalise invariances in the neighbourhood. To implement Conway's life, a binary automaton with an 8-neighbourhood requires 512 separate associations rather than three simple rules in CAM. We need a scheme that reduces this to a more manageable number for more complex systems. Smolensky [26][27] and researchers in Queensland [25] worked on schemes for binding variables (represented as vectors or tensors) and this provides a technique to solve the difficulty.

figure 9 rotational symmetry is not recognised with the current training scheme



The techniques in [25] indicate a way to give inputs to an associative memory a context by creating a tensor from all vectors relating to contextually linked information which allows a recall given part of this information. This might be extended to linking a pattern on the input with some contextual information that gives some clue as to the symmetry that is present within the rules. Recalling of the patterns should be a comparatively simple exercise compared to the training.

8.4 Other neural network based implementations of cellular automata

Goles and Martinez talk at length on cellular neural systems in their book [50], as do the proceedings of CNNA-94 [refs.]

9 Conclusions

The literature indicates that cellular automata are highly parallel and flexible systems that are ripe for implementation on suitable parallel hardware or composite hardware and software systems. It also indicates that they may be implemented using neural systems as the state machines. Associative memories offer a good prospect for implementation of state machines since they allow simple associations which is what is required for a state machine, their properties are well known, they are fast and simple and are an active research area offering techniques for enhancement of performance, provided their defences, i.e. possible inexactitude of recall, are dealt with.

10 References

[1] ?, “Artificial Life”,?[

[2] S M Ulam, R G Schrandt, “On Patterns of Growth of Figures in Two Dimensions”, Notices of the American Mathematical Society, Vol. 7, p642-, 1960

Concerned with generation and growth of figures using CA like rules. The Ulam and Schrandt papers are generally good at using graphics and photographs but the prose style is a little dull compared to more recent reviews of the subject and their work.

[3] S M Ulam, “On Some Mathematical Problems Connected with Patterns of Growth of Figures”, Proceedings of the Symposia in Applied Mathematics, Vol. 14, p215-224, 1962

[4] E F Codd, “Cellular Automata”, Academic Press, 1970

This is considered to be a seminal work and helped to establish a firm mathematical language for describing transition functions

in CA (others have looked at higher level structures, notably Holland). The quantity of mathematics makes this hard going, especially given that some is inadequately explained, notably the equations concerning subconfiguration early on. Worth perservering with.

[5] A W Burks, “Von Neumann’s Self-Reproducing Automata”, Essays on Cellular Automata, edited by A W Burks, University of Illinois Press, 1970

[6] E F Moore, “Machine Models of Self-Reproduction”, Essays on Cellular Automata, edited by A W Burks, University of Illinois Press, 1970

A description of synchronous, homogeneous cellular automata in abstract terms. Some interesting discussions on some general aspects, but mostly actually a discussion of tessellation structures.

[7] A W Burks, “Automata Models of Self-Reproduction”, Information Processing, The Information Processing Society of Japan, 1966

[8] A W Burks, “Programming and the Theory of Cellular Automata”, Essays on Cellular Automata, edited by A W Burks, University of Illinois Press, 1970

The implementation of Turing machines in CA. Important since it discusses how to implement general computations in CA by modifying the transition functions, albeit in theoretical terms. Rather too much (necessary) maths and hard going.

[9] A W Burks, “Towards a Theory of Automata Based on More Realistic Primitive Elements”, Essays on Cellular Automata, edited by A W Burks, University of Illinois Press, 1970

The discussion of tapes etc. seems quaint, although based in Turning’s terminology. Briefly discusses probablistic automata. Deals much with actual low-level physical implementation of cellular automata using basic electronic components which is possibly not totally relevant now, until a reversible optical CA is used as a thermodynamically efficient general processor (as Toffoli would have it).

[10] J W Thatcher, “Self-Describing Turing Machines and Self-Reproducing Cellular Automata”, Essays on Cellular Automata, edited by A W Burks, University of Illinois Press, 1970

Good discussion of Turning machines but does not really much discuss cellular automata so title a misnomer.

[11] J W Thatcher, “Universality in the von Neumann Cellular Model” Essays on Cellular Automata, edited by A W Burks, University of Illinois Press, 1970

Discusses global transition functions using slightly different terminology to Codd, which is less familiar but perhaps clearer when discussing subconfigurations. Important points are the discussion of pre-conditions required for self-reproduction thus outlining which classes of CA are self-reproducing. Indicates that self-reproduction mutually implies computing universality. Some parts concerning von Neumann’s system is rather complex.

[12] A M Turning, “On Computable Numbers with an Application to the Entscheidungsproblem”, Proceedings of

the London Mathematical Society, Series 2, Number 42, p230-265, 1936

[13] K Preston, M J B Duff, “Modern Cellular Automata - Theory and Applications”, Plenum Press, 1984

Fairly nitty-gritty look at “cellular logic” which is based mostly on the work of Ulam and Schrandt and is similar to the theory of convolution kernels when applied to image processing. Much of the book is concerned with image processing and really requires a knowledge of this. Good historical review of early attempts to build CA in hardware

[14] K Preston, M J B Duff, et al., “Basics of Cellular Logic with Some Applications in Medical Image Processing”, Proceedings of the IEEE, May 1979, p826-85, 1979

[15] K Preston, “Cellular Logic Arrays for Image Processing”, Handbook of Pattern Recognition, p395-436, Academic Press, 1986

[16] G Weisbuch, “Simulated Annealing”, Complex Systems Dynamics, G Weisbuch, Santa Fe Institute, 1990

Describes image processing as a form of cellular based simulated annealing. A more general overview of such techniques than Kendall and Duff. Considered a seminal work.

[17] G Weisbuch, “Cellular Automata”, from above

Good and easily readable overview of cellular automata without much in the way of mathematics and plenty of diagrams.

[18] T Toffoli, N Margolus, “Part III: Physical Modelling”, Cellular Automata Machines, T Toffoli and N Margolus, MIT Press, 1985

Perhaps over full of information and examples for an easy read, but a mine of ideas should one be interested in using a cellular automata for modelling a physical system. Toffoli suggests that differential equations may be directly modelled by CA. Easily readable.

[19] T Toffoli, N Margolus, “Cellular Automata Machines”, MIT Press, 1985

Much information is related to the specific CAM-6 implementation which obscures some of the theoretical points early on. An easy read and a good companion to Weisbuch.

[20] R K Squier, K Steiglitz, “Programmable Parallel Arithmetic in Cellular Automata Using a Particle Model”, 1994

An interesting use of CA to perform arithmetic by using to CA to model particles travelling down a pipe and interacting in a way reminiscent of quantum physics. The state functions are implemented in a traditional way. In some ways a curiosity although some work a York is implemeting a 2d associative memory system that bears some similarities.

[21] J Von Neumann, “The General and Logical Theory of Automata”, Cerebral Mechanisms in Behaviour, Wiley, 1941

An early paper on automata of the Ulam and Schrandt form. A little outdated.

[22] J Von Neumann (edited A W Burks), “The Theory of Self-Reproducing Automata”, University of Illinois Press, 1966

Seminal early work. Burks’ style is rather dry and betrays his history as a developer of early electronic computers. Considered a classic because it was first.

[23] J Von Neumann, “A System of 29 States with a General Transition Rule”, The Theory of Self-Reproducing Automata, University of Illinois Press, 1966

More of the same

[24] W Hasselbring, “Programming Cellular Automata for Image Processing”, Presented at BCS-PPG Meeting on Cellular Automata, Imperial College, 1992

More of a technical report than a paper. Describes a particular instance of a language to interface between a user and a cellular automata (and with nothing particularly related to image processing in this paper). Interesting as a spark for ideas and an inspiration to do something similar but only a curiosity since there is little in the way of a standard for such languages.

[25] J Wiles, G S Halford, J E M Stewart, M S Humphreys, J D Bain, W H Wilson, “Tensor Models: A creative basis for memory retrieval and analogical mapping”, Technical Report no. 218, Key Centre for Software Technology, Department of Computer Science, University of Queensland

Easily readable if long technical report. Good for ideas concerning variable bindings for context linking, and backs up Smolensky’s papers.

[26] P Smolensky, “Tensor Product Variable Binding and the Representation of Symbolic Structures in Connectionist Systems”, Artificial Intelligence, 1987 IEEE Conference on Neural Information Processing Systems, 1987

A classic paper. The principles are clear if the mathematical justification and derivation is not at first. The ideas seem sound and are much used. Important for suggesting that similar principles may be applied to binary tensors.

[27] P Smolensky, “Tensor Product Production System: A Modular Architecture and Representation”, Connection Science, 1(1), p 53-68, 1989

[28] R C Paton, “Some Computational Models at the Cellular Level”, Biosystems (1993) 29, p 63-75, 1993

A chatty paper that suggests some ideas concerning CA, synchronicity, and emergent behaviour. Many of the analogies are biological and it seems aimed at an audience of biologists. SOmetimes feels that the ideas, while good, are not necessarily backed up by a sound literature review.

[29] J Von Neumann, “Probablistic Logics and the Synthesis of Reliable Organisms from Unreliable Components”, Automata Studies edited by C E Shannon and J McCarthy, Princeton University Press, 1956

A seminal paper. Describes a possible sub class of automata that can repair mistakes made by one element, hence biological analogies. Very important for those using imperfect associative memories as the heart of CAs to do any sort of symbolic logic.

[30] J Austin, “The design and application of associative memories for scene analysis”, YCST 87/06, University of York, 1986

[31] J Austin, “ADAM: A Distributed Associative Memory For Scene Analysis”, Proceedings of First International Conference on Neural Networks, vol. IV, 1987

[32] J Austin, M Brown, S Buckle, A Moulds, R Pack, A Turner, “The Cellular Neural Network Associative Processor”, YCS 242 (1994), University of York, 1994

[33] J Austin, “Uncertain Reasoning with RAM based Neural Networks”, Intelligent Systems, Vol. 2, nos 1-4, p 121-154, 1992

[34] J Austin, “Distributed Associative Memories for High-Speed Symbolic Reasoning”, International Journal on Fuzzy Sets and Systems, 1995

[35] J Austin, “High Speed Invariant Pattern Recognitions Using Adaptive Neural Networks”, IEE Proceedings of Conference in Image Processing and its Applications, Warwick, p 28-32, 1989

[36] J Austin, “An associative neural architecture for invariant pattern classification”, Proceedings of first IEE International Conference on Artificial Neural Networks, London, p196-200, 1989

[37] J Austin, “Parallel Distributed Computation”, Proceedings of International Conference on Artificial Neural Networks, Brighton, 1992

[38] J Austin, I Kelly, “Hardware Support for Distributed Associative Memories”, WNNW Workshop Proceedings, York, 1993

[39] J Austin, J Kennedy, “A Hardware Implementation of a Binary Neural Network”, MicroNeuro 1994

[40] J Austin, “Correlation Matrix Memories for Knowledge Manipulation”, International Conference on Neural Networks, Fuzzy Logic and Soft Computing, Iizuka, Japan, 1994

[41] J Austin, “The cellular Neural Network Associative Processor”, Proceedings IEE Conference on Image Processing and its Applications, 1995

[42] J Austin and J Kennedy, “The Design of a Dedicated ADAM Processor”, Proceedings of IEE Conference on Image Processing and its Applications”, 1995

[43] J Austin, B Cass, J Kennedy, “A Hardware Implementation of a Binary Neural Image Processor”, 1995

[44] J Austin, “Statistical Parallelism”, 1995

Useful paper giving bare outline of the idea. Easily readable. Does not touch on potential pitfalls.

[45] J Austin., T Macek, G Morgan, “A Transputer Implementation of the ADAM Neural Network”, 1995

An overview of a particular implementation, an alternative to the SAT processor.

[46] T Kohonen, “Associative Memory: A System-Theoretical Approach”, Springer-Verlag, 1978

[47] D J Willshaw, “Holography, Associative Memory and Inductive Generalisation”, Parallel Models of Associative Memory, edit by G E Hinton and J A Anderson, Lawrence Erbaum Associates, 1989

[48] T Kohonen, “Storage and Processing of Information in Distributed Associative Memory Systems”, Parallel Models of Associative Memory, edit by G E Hinton and J A Anderson, Lawrence Erbaum Associates, 1989

[49] G E Hinton, “Implementing Semantic Networks in Parallel Hardware”, Parallel Models of Associative Memory, edit by G E Hinton and J A Anderson, Lawrence Erbaum Associates, 1989

[50] E Goles, S Matinez, “Neural and Automata Networks”, Kluwer Academic Publishers, 1990

[51] S Wolfram, “Universality and Complexity in Cellular Automata”, Physica D, volume 10, 1984

General Comments: overlong. plenty of pictures. lots of maths.

[52] T E Ingerson, “Structure in Asynchronous Cellular Automata”, Physica D, vol. 10, 1984

Good review of Wolfram’s work on this paper. Short, easily readable. Poor graphics.

[53] M Gardener, “Mathematical Games”, Scientific American, 223, no 4, 1970

[54] N Margolus, “Physics like models of computation”, Physica D, vol. 10. 1984

General comments: long. easily readable. possibly labours points. over enthusiastic language!

[55] G Y Vichniac, “Simulating Physics with CA”, Physica D, vol. 10, 1984

Generally readable paper if a little overdetailed at times - looks at examples too much as compared to concepts. long

[56] T Toffoli, “CA as an alternative to DE in Modelling Physics”, Physica D, vol. 10, 1984

Quite complex language - leaps in very soon and requires a knowledge of physics. Maths from page 1. Heavy going. Important paper in that it makes the point that in general the only way to be absolutely sure of the behaviour of a CA is to run it.

[57] S Omohundo, “Modelling CA with partial differential equations”, Physica D, vol. 10, 1984

[58] C G Langton, “Self-reproduction on CA”, Physica D, vol. 10, 1984

Good background to problem. i.e. what is self-reproduction? Universal construction’ excludes humans from being alive. Concept

of active self-reproduction. Discusses own system (detailed and complex)

[59] S A Kaufmann, “Emergent Properties in Random Cellular Automata”, Physica D, vol. 10, 1984

review of well known results - a good paper. not too much maths. longish.

[60] T Toffoli, “CAM: A high performance cellular automata machine”, Physica D, vol. 10, 1984

Detailed description of a particular implementation, but readable.plenty of colour pictures

[61] K Preston, “4d logical tranforms: data processing by cellular automata”, Physica D, vol. 10, 1984

i.e. 3d image analysis. A bit obscure

[62] W D Hillis, “The connection machine: a computer architecture based on cellular automata”, Physica D, vol. 10, 1984

Similar to CAM in some ways - uses CA as a general processor with own idiosyncratic language for representing relationships. claims not to be associative memory, systolic array or database machine. interesting.

[63] H de Garis, “CAM-BRAIN: The Genetic Programming of an Artificial Brain Which Grows/Evolves at Electronic Speeds in a Cellular Automata Machine”, Proceedings of the 1994 IEEE World Congress on Computational Intelligence, pp 337-339

[64] “H de Garis, “An artificial brain (ATR’s CAM-BRAIN project aims to build/evolve an artificial brain with a million net modules inside a trillion cell cellular automata machine“, New Generation Computing, vol. 12, p 215-221, 1994)

A fascinating paper covering a rather complex hardware implementation of neural network on a cellular automata, with details of the theoretical aspects and the immense amount of work that has gone into it. Given the amount of work still to do by the official target date of 2001 I can’t help feeling that de Garis is a mad scientist.

[65] J Kari, "Reversibility of 2D Cellular Automata is undecidable", Physica D, vol. 45, 1990

[66] C Bays, "Candidates for the Game of Life in Three Dimensions", Complex Systems, vol. 1, no. 2, 1987

[67] P Gacs, "Reliable Computation with cellular automata", Journal of Computer System Science, vol. 32, no. 1, 1986

[68] P Gacs, "Self-correcting two-dimensional arrays", Randomness in Computation edited by Silvio Micali, JAI Press, 1989

[69] H Gutowitz, "Statistical Properties of Cellular Automata in the Context of Learning and Recognition", Learning and Recognition - a Modern Approach edited by K H Zhao, World Scientific Publishing, 1989

[70] K Culik, S Yu, "Undecidability of Cellular Automata classification schemes", Complex Systems, vol. 2, 1988

[71] H Hartman, G Vichniac, "Inhomogeneous cellular automata", Disordered Systems and Biological Organization edited by E Bienenstock et al, 1990

[72] M Mitchell, JP Crutchfield, P T Hraber, "Evolving Cellular Automata to Perform Computations", Physica D, 1994

[73] Z Aleskic, "Computation in Inhomogenous Cellular Automata", Proceedings of ANU-94, 1994

[74] F C Richards, et. al., "Extracting Cellular Automata Rules from Experimental Data", Physica D, vol, 45, 1990

[75] G Vichniac, P Tamayo, H Hartman, "Annealed and Quenched Inhomogeneous Cellular Automata", Journal of Statistical Physics, vol. 45, 1986

[76] J Myhill, "The Converse of Moore's Garden-of-Eden Theorem"