WORKING NOTE AC94

Alan Creak
6 January 1995

TWO-STAGE AUTOMATA

This discussion is based on several publications of J.P. Crutchfield, M. Mitchell, and various others, here referred to generically as C&M. It was developed from discussions with Paul Qualtrough and David Norman in the early months of 1994.

C&M are concerned with the development of cellular automata which solve problems, in the sense that they transform initial configurations into predefined final configurations selected according to some predicate satisfied by the initial configurations. They therefore act as classifiers of the initial configurations. The major problem addressed by C&M is the *majority problem*, and they require their automata to proceed to an overall state in which each cell is in the state which was in the majority in the initial configuration. They use two-state cells; I shall call the states black and white. The behaviour of an automaton is determined by the behaviour of its cells, all of which are identical. Each cell changes its state to a new state determined by its own state and the states of its neighbours; C&M's cells can see a neighbourhood of seven cells, including themselves. The possible changes of state of a cell are described by a list of the possible configurations of the effective neighbourhood, with the corresponding cell state after the transition defined for each; this list is called the *rule* for the automaton.

An interesting fact about C&M's work is their use of genetic algorithms to develop the rules. These algorithms use the principles of natural selection to evolve increasingly effective solutions to the problem.

BACKGROUND.

The idea behind the suggestions put forward in this note is that trying to achieve several things with limited resources is likely to lead to conflict, so it should be easier to recognise that you're doing *N* things, and to learn *N* jobs separately. (I'll assume $N = 2$, but that isn't necessarily a general rule.)

This notion impressed itself upon me in the context of cellular automata while I was brooding on the severe constraints imposed on the rule of such an automaton by the requirements of complex behaviour. Using C&M's parameters, the whole behaviour must be represented in 128 bits. Consider the constraints imposed by a requirement that a black-white boundary separating at least two blacks on the left and two whites on the right must be preserved :

(⌀ means don't care.) For each of the cells with a defined value in the new pattern, eight bits in the rule must be defined; for example, to guarantee that the first ● will be preserved, its cell (and therefore all the cells of the automaton) must include in its behaviour the transitions :

with all eight possible permutations of ○ and ● in the don't-care positions. To preserve the boundary as required, there are therefore 32 constraints, all of which must be represented in the 128 bits of the rule, so four requirements of this sort would completely define the rule – or clash, in which case they would be impossible to satisfy simultaneously.

In fact, this is a fairly fierce restriction because of the "don't-know"s, and it may take fewer to propagate a simple pattern. For example, for the domain :

which appears in some of the C&M pictures[1] there are only four constraints. I have no idea what one should consider to be a representative number of constraints for a pattern, but it seems likely to be several, and in a space of 128 bits there is soon going to be congestion if the automaton is intended to exhibit a lot of complicated behaviour.

Observe, incidentally, that arguments based on the enormous search space of 2^{128} rules "so surely there *must* be an answer somewhere" are beside the point. The constraints drastically limit the search space, and once there are enough constraints to determine each of the bits in the rule, the size of the

search space is 1 – and if two constraints conflict, the search space size is 0, no matter how few transformations are required.

This sort of argument suggests that it would be hard to develop a rule to do a clearly defined job in which the behaviour of a cell was determined by a predefined pattern of behaviour of the automaton as a whole – but also that it would be even harder to develop a rule which would make the automaton do two quite different jobs. Such a requirement might also impose too many constraints on the formation of the rule, by requiring that some cell input leads to contrary outputs at different stages of the overall process.

This doesn't necessarily mean that there is no rule which makes the automaton behave as desired. My argument is based on first imagining a mechanism which would produce the desired behaviour, then, knowing just what transformations we want the cells to perform, trying to encode those transformations in the rule. It doesn't eliminate the possibility that there may be other transformations which cause the desired behaviour. C&M's approach is to give a specification of the desired final state for – in effect – all the initial states, and to rely on a genetic algorithm to find a rule which will do the job. What my argument does suggest is that 128 bits isn't very many to encode a number of coherent functions, so that simple and obvious (to us) methods might be ruled out by sheer congestion. In terms of the "engineering" model[2], we prevent the algorithm from experimenting freely by restricting it to a very limited set of apparatus and forcing it to conduct several experiments at the same time. (I think that is a reasonably good analogy which continues the "engineering" model, but am open to argument.)

THE PROPOSAL.

Looking at the pictures in one of the C&M papers[1], it seems that in many cases there are two stages to a solution. In the first stage, the disorderly initial configuration is classified, and probably simplified, in some way, while in the second stage the results from the first stage are propagated through the whole automaton. There is an analogy here with human reasoning : first we find a way to express a problem in formal terms, then we use the available formal methods to work out the solution.

The same notion appears (not entirely independently) in my note on cellular automata which are intended to perform calculations[2], but with the addition of speculation on the nature of the two stages of operation. I conjecture that the major calculation in C&M's automata is carried out by logical processes depending on the expression of certain predicates about the configuration in the form of spatial patterns; but before this can happen, the initial configuration must be encoded into these patterns.

Perhaps, then, it would be interesting to investigate the properties of *two-stage automata*, with separate rules for initial encoding and subsequent argument. That proposition is explored in the rest of this note.

PRECEDENTS.

If this is such a good idea, can we find any examples ? I think we can.

The first example which occurred to me was (as mentioned in the preceding paragraph) human reasoning. I was looking at C&M's pictures with an eye biased by recent thoughts about the logical processing performed in the later stages of their computations. I believed that I had discerned certain logical operations in the later development[2], but these seemed to depend on the association of certain predicates with regular patterns extending over a comparatively long sequence of cells, and I didn't know where the long sequences came from. This drew my attention to the first few cycles of the automaton's operation, where somehow a quite arbitrary initial configuration was hammered into something more orderly by an obscure process. Having nothing to go on but C&M's pictures, I didn't get very far in working out what the process was, but it did seem reasonable that some sort of local grouping sequence should precede the more global computation which followed.

There is further evidence of the same principle in the design of machines. In a sense, that isn't new, as machines are also among the fruit of human reasoning, but the observation does show that the principle works in mechanical devices, not just in thinking. Many machines have distinct starting modes and running modes, and almost all complicated machines have a set of discrete controls which change their machines' operating modes, sometimes quite considerably, between mutually exclusive operating modes. The separate modes may be implemented by common components, but the economy of sharing seems to give way rather soon to the simplicity of separate components to handle the different tasks. The

combination photocopier and kitchen stove may catch on some day, but if it does I doubt whether the hot parts will be shared.

It is fashionable nowadays to assume that if anything can be done, nature will have done it by evolution. That's nonsense, but it's still interesting to survey the field, and in this case it does seem to be populated to some extent. In the case of animals, I can think of examples including egg-chicken, egg-caterpillar-chrysalis-butterfly, migration and hibernation, and – almost universally in bigger animals ? – embryo-child-adult. (There are also the remarkable parasites which inhabit several different species during their lifetimes, but I've forgotten the details. Liver flukes ?) Vegetables do it too, I'm sure, but all I can think of just now are lancewood and kauri, both of which have growing forms different from the adult species. In more technical terms, the common genotypes express themselves as different phenotypes. I'm not sure that such details are terribly helpful in this discussion, but the point is that it seems to happen.

It's interesting that in both biological and mechanical contexts one can discern two varieties of behaviour. In some cases, the switch between modes is quite abrupt, triggered by some profound biological event or external stimulus; in others, the transition is more gentle, and looks more like the same agent adapting to a different environment.

ORGANISATION.

The major difference between a two-stage automaton and the sort investigated by C&M is the transition between the two stages. What sort of trigger initiates the switch, and how does it act ?

There are several possible triggers. Perhaps the simplest is some signal from an external agent (such as a person). This does not appeal much, either as a practical mechanism for getting results (because an autonomous device is clearly easier to use, and likely to be more reliable), or as a contribution towards the development of an understanding of the evolution of cooperative systems (because it sidesteps the important point).

A rather more satisfactory trigger is the age of the computation, measured in steps of change since the computation began. There is an unattractive arbitrariness about predefining some number of cycles after which the behaviour will change, but at least it permits the construction of an autonomous automaton. It also removes the need for a global agent, for a counter can be built into each cell. (As in all these examples, I preserve the global timebase.) But, of course, I don't know how to define the correct number of steps. My guess for the majority problem is that the number is quite small, because I think that a plausible first stage is "just" a matter of tidying up local arbitrary patterns into something a bit more convenient. More generally, perhaps the number should be regarded as a parameter to be determined along with the cells' transformation rule as the automaton is developed. If we follow C&M by using a genetic algorithm, then the trigger time should be chosen by the algorithm too. There is a sense in which this choice is deeper than the determination of the rule which the cells use at every iteration cycle, but it is certainly part of the design of the system, and that is not unreasonably dependent on the system's genetic material.

A trigger which seems to be in some sense more natural, in that it relies most strongly on the ordinary mechanisms of the automaton, relies on the appearance of some pattern in a cell's field of view to effect the switch. This is "natural" in the sense that all the cell's operations now depend on what it sees, and there is no mysterious external agent, nor any need for a built-in counter – and also in that the changeover is precipitated by the state of readiness of the automaton itself, which is surely more appropriate than the other methods. In an automaton of this sort, the switch could occur at different times in different parts of the system. That is not obviously either good or bad.

It seems possible that the existence of different sorts of mode-switching which I noted in the last paragraph of the preceding section is to do with different sorts of trigger – purely time-dependent or external triggers for the abrupt changes, but state-dependent triggers for the more gradual adaptations.

A SAMPLE PROBLEM.

Here is an example to illustrate what I mean. It is a variety of the majority problem, carefully cooked to be solvable in two stages. I can't think of a reliable one-stage solution. It uses a linear array of cells, not a ring (because I can make it work that way).

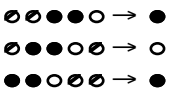
Each cell has two states, black and white, and a radius of 2.

The **initial state** has all the blacks at one end (say, left) and all the whites at the other. (You could make a three-stage problem by beginning with arbitrarily distributed blacks and whites and adding a first stage in which the blacks and whites are collected together, but it adds rather little to the discussion.)

The **final state** is all black (white) if there is a black (white) majority, and otherwise grey.

Here's a two-stage solution :

Stage 1 : Change ●●○ to ●○●. That will spread out the blacks to make a 1:1 grey, but no further. If there's an excess of blacks or whites, the unpaired set will finish up at the right hand end. Continue until the trigger for the change of rule. The radius is big enough to make the transformation work, as all necessary cases can be recognised within a radius of 2 cells. The list of transformations is :



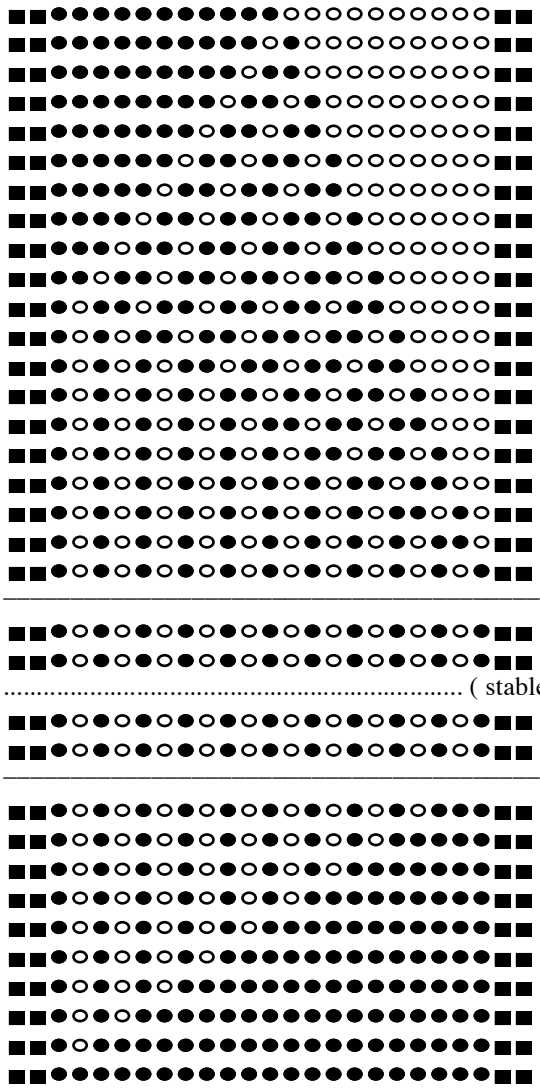
The state of a cell in any other context remains unchanged. The same is true of the first transformation illustrated; so in the complete rule of 80 transformations, only 15 result in a change of state. (End effects are included.)

I think that the **trigger** must be a timed trigger, because I can't think of a pattern, or set of patterns, recognisable within radius 2 which is, or are, characteristic of the end of the first stage. Inspection of the diagram below shows that the changeover time must be something like twice the size of the automaton.

Stage 2 : Change ●○○ to ○○○, and ○●● to ●●●, and ○●■ to ●●■, where ■ represents the end of the automaton. This will propagate the excess back from the right hand end to fill the whole array. Continue until the process must have finished. The sequence ○●■ must be included to deal with the case of an automaton with an odd number of cells, and just one more black than white. The final sequence ●○■ will occur when an equal number of cells are initially black and white; as this contains none of the patterns changed in stage 2, the uniform grey state will remain unchanged.

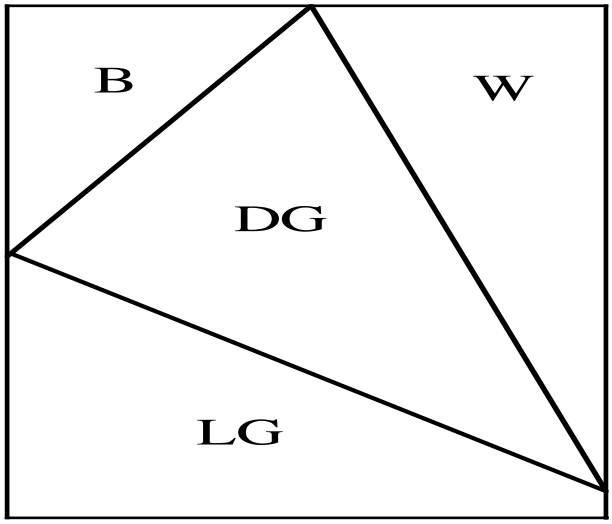
There's an example of a run of my proposed problem on the next page. It's all done by hand, but honest. The final line of stage 1 shows correctly (by the black cell at the right-hand end) that there were more blacks than whites in the original line. The final pattern is stable, and would still be stable if more white cells were added to the right hand end of the automaton – but then the rightmost cells would be white, showing that (except for the case of equal numbers of blacks and whites) there would then be a white majority.

Several features of this diagram are intriguing, and some are parallel to the corresponding features observed in the C&M work; discussion follows the diagram.



"PARTICLES".

The "particle" tracks for stage 1 of the computation are shown on the diagram below. They are the boundaries between regions of different patterns.



As in the C&M work, the particle tracks have different gradients; the gradients of the B-DG, W-DG, and LG-DG are, respectively, –1:1, 2:1, and –1:2. The LG-W boundary which would develop if there were more white cells is vertical.

REGIONS.

There are five regions in the diagram, all appearing as different patterns, except for the top left region labelled B and the bottom right (stage 2) region, both of which are completely black. I suggested elsewhere[2] that each region represents some predicate which is known about its cells, and that the "particles" were better regarded as traces of the execution of logical operations between these predicates. This interpretation works to some extent with these patterns.

First consider the initial phase, in which both B and W regions are present. During this phase, the B (W) area means "All these cells were black (white) in the initial configuration", and the DG area means "In the initial configuration, there were twice as many black cells as white in this area". It is quite exciting that, without any preplanning, the pattern developed for each of the areas satisfies its predicate. That's what I expected (though the 2:1 ratio was a surprise), but very rewarding to see in action.

Once one of the original B or W regions disappears, these interpretations are no longer sustainable. For example, once the B region in the diagram has been consumed, the DG region must change its meaning to something like "I need this many 2:1 blacks and whites to balance up with the 1:1 LG area on the left to give the right total over our combined region"; and the LG area means "This area is evenly balanced, so don't count it in the final result". This is less æsthetically satisfactory – but the process still works.

It works, because it was designed to work. The simple transformation of stage 1 directly implements the self-evident observation that if we keep on splitting up adjacent pairs of blacks by moving one black to the right whenever possible we will eventually end up (if there's no restriction to the right) with a uniform 1:1 grey (LG) pattern. The 2:1 DG pattern was no part of the original plan; while its appearance is clearly a natural consequence of the algorithm in hindsight, it was quite unexpected. It is therefore not too surprising that I struggle to find an interpretation of its interactions with its neighbours. Unsurprising or not, though, it remains disconcerting; to attain what seems to be a simple and straightforward goal, one expects to use simple and straightforward mechanisms with easily understood effects from which one can demonstrate that the goal must be attained. To attain a simple and straightforward goal by inscrutable means jars one's susceptibilities.

PARTICLES.

If we can't understand the logic of the regions, do the particles make more sense ? Not obviously. For example, the relationship between B, W, and DG in the initial phase of the computation only works because the two "particles" bounding the DG move freely throughout. For the whole system, the blacks and whites are being mingled to produce, in this early phase, a 2:1 mixture. But what is each of the particles doing ?

There is obviously some low-level mechanistic statement of the behaviour of each particle, amounting to a list of all the cell transitions whcih make it work, and no doubt if the statements describing the two particles were written down then the consequence – the development of the 2:1 region – could be derived somehow. Once again, though, this all seems to be very remote from the simple design principle of the system.

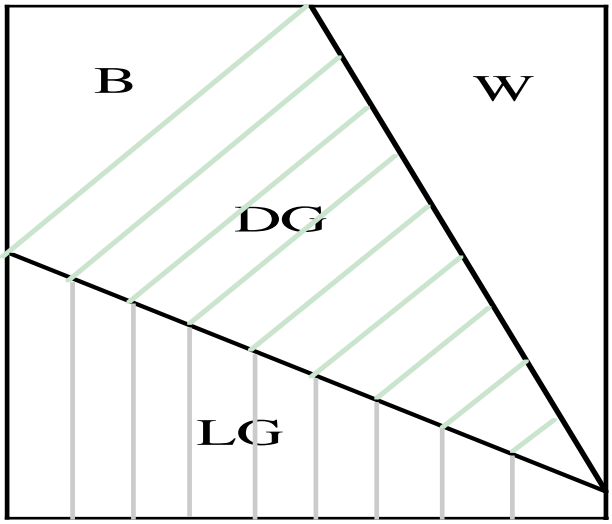
I have tried to identify a sensible (higher level, comprehensible) logical operation which can be identified with either of the particles separately, and have failed.

WHAT ARE THE PARTICLES ?

Before leaving the particles, it is interesting to ask how we know what they are. C&M identify particles with boundaries between regular regions with different patterns, and so far I have done the same. On the other hand, C&M also regard particles as a means of transmitting information from place to place in the automaton. It is by no means clear that these two views coincide.

I emphasised the second view in my previous note[2], where I suggested that particles could carry quantitative information from place to place through the automaton. What I didn't notice then was that some of these particles were, in effect, boundaries between *identical* regions. There are some on show in our specimen computation.

Inspect the DG region. Until now, I have regarded it as a homogeneous region of the diagram, but it could equally be seen as a region in which a succession of white particles moves into the black area. Each diagonal white line is one of the particles. Similarly, the LG region can be seen as a set of stationary white particles. The white particles are indicated schematically (in grey) in the diagram below.



A very interesting feature of these particles is that they have very simple meanings : each white particles means "Here is a white cell". This meaning is consistent, it corresponds exactly to the design of the automaton, and it works.

What about the C&M sort of particle ? They acquire a rather different meaning, as they now in effect operate on particles to do useful things. The DG-W boundary can now be seen as a glider gun[3] which has the special property of conserving the number of black and white cells. The LG-DG boundary is a sort of store, where the particles are stacked in the form of rather different particles, but again with conservation of numbers of cells with different states. These statistics-conserving operators are just what we need to perform the precise calculations discussed earlier[2].

Is this a better way of looking at the computation ? It seems to be much more comprehensible than C&M's way, and perhaps there is some promise of constructing a library of operators.

STAGE 2.

Stage 2 is simpler, having only one "particle" separating a 1:1 area from an area which means that there's a majority of (in this example) black. The logic associated with this particle is good, and stands by itself : any majority can correctly absorb a 1:1 area.

WHY DO WE NEED TWO STAGES ?

Suppose we choose a different encoding for the final state : the majority colour is shown in the rightmost cell at the end of the run. Then we don't need stage 2 at all.

Does that mean that the two-stage requirement is really trivial ? Can it always be eliminated by reformulating the final encoding ?

I don't think that the question is sensible. Encoding is, after all, just another sort of computation; it's only our perception of it (or maybe that we can perceive the answer we want at the end of stage 1) which tempts us to think otherwise. (We can equally perceive the answer at the beginning of stage 1 – just look at the colour of the middle cell.) There's a sense in which this is always so. Quite generally, the result of any mathematical or logical operation is implicit in the original state, or we wouldn't be able to work it out; all we ask for is a particular transformation, and if it turns out to need a two-stage calculation then that's an interesting property of the system.

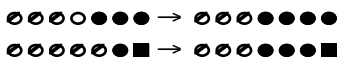
But pouring scorn on a question doesn't answer it, and I still haven't demonstrated that two stages are necessary in principle, or that one stage will do if we reformulate the problem. On the first point, the answer is evidently no : some requirements can be met very well by a one-stage automaton. (If we really want the solution to appear in the right-hand cell, we know how to do it in one phase.) The second point

is less simple, because it isn't well posed. What do we mean by "reformulating the final encoding" ? I haven't defined it; but from the context it must mean something like "choosing any final encoding attainable by a one-stage automaton from which the required result is self-evident". Now we have to define "self-evident". I shall go no further with this "analysis", and assert that there is no simple answer to the question. What I have shown is that two stages can sometimes be effective, and I am (for the moment) content with that.

A UNIFIED AUTOMATON ?

For a while I thought that I could devise a rule which would solve the problem in one go – that is, without a change of rule. It turns out that my bright idea falls well short of a general method, but the way it half works is interesting.

Looking at my first sketch of the picture on page 1, I was struck with the behaviour of the stage 2 part. This is very simple, requiring only that a solid black area at the right boundary should propagate left to fill all the cells. It occurred to me that we could manage this quite automatically by adding new rules of radius 3 which implement these transformations :



The first of these transformations recognises that the appearance of any clump of three or more blacks to the right of a white implies an excess of blacks, so the black area can be propagated to the left. ("Three or more" is essential, because clumps of two blacks occur throughout region DG; a rule of radius three is essential to give a sufficiently broad view of the neighbourhood to the cell which must change its state. This cell must be able to see its own state (white) and those of its three neighbours to the right (black).) The second covers the special case in which the excess of black over white is less than three, so that a sequence of three blacks can never be developed, and explicit recognition of a single black at the right boundary is necessary.

It is interesting to notice two points relating to the second transformation. First, it is only required for the special case, so if it were known that the excess of blacks over whites would never be less than three, the first transformation alone would suffice. If the second transformation is included, though, it is always this transformation which initiates the expansion of the black region to the left.

The corresponding transformation for a white majority must recognise the vertical boundary between the LG and W regions which develops once the DG region has disappeared, and distinguish this from any pattern which can occur at the boundary between DG and W. The minimum transformation appears to be :

A symmetrical automaton once again requires a rule of radius three to effect the transformation. A single case will suffice :

(The "don't care" caters for the possible appearance of the automaton boundary.)

A GENERAL METHOD ?

How have I managed to construct this two-stage automaton with an automatic trigger ? In this case, the trick is worked by this sequence of operations :

1 : Devise a two-stage solution using rules which are as simple (small radius) as possible.

2 : Find patterns which are characteristic of all possible final states of the first stage, and use these to generate further patterns which can be recognised by the rule of the second stage.

3 : Mix them all together, and let it go.

There's certainly no reason to believe that this recipe will always work – which is a shame, as Paul pointed out, for a means of synthesising cellular automata to perform specified computations would be really exciting.

Worse, there are reasons for expecting that the recipe would not work in certain cases. Perhaps the most obvious is a scheme in which some pattern of cell inputs was required to give different outputs in the two stages. The example which I have discussed is just such a case, for the pattern ●○●○○ must lead to ● in the first stage, and ○ in the second. I have avoided this complication in my description by extending the radius of the cells' input ranges, but this is not necessarily generally possible. Similar comments apply to the suggestion that more stages may be added to implement more complex proposals.

That brings us back to my earlier question : why do we need two stages ? The answer seems to be that we need the separate rules to ensure that the mechanisms of the stages don't clash. Clearly, if we are allowed to change the rule of the automaton completely at the trigger rather than try to devise a simple automaton, many of the constraints which I have mentioned disappear. All we need now is a way to implement the operation of the first stage, and a way to recognise when it is complete which we can use for a trigger – and, if we are content to use a timed trigger, we don't even need that.

REFERENCES.

1 : R. Das, M. Mitchell, J.P. Crutchfield : *A genetic algorithm discovers particle-based computation in cellular automata*, preprint for the Third Parallel Problem-Solving from Nature Conference, March, 1994.

2 : G.A. Creak : *Cellular automata with a purpose*, unpublished Working note AC93 (December, 1994).

3 : A. Wuensche : *Complexity in one-d cellular automata; gliders, basins of attraction and the Z parameter* , Cognitive Science Research Paper 321, University of Sussex, 1994.