The Beehive Cellular Automaton with Memory

RAMÓN ALONSO-SANZ

ETS I Agrónomos (Estadística), C.Universitaria. 28040, Madrid, Spain.
Grupo de Sistemas Complejos-UPM.
E-mail: ramon.alonso@upm.es

Received: March 01, 2006.

The major features of conventional cellular automata include fixed topology and absence of memory. The effect of simple memory (memory in cells and links) on the two-dimensional, three-state beehive cellular automaton is explored in this paper.

Keywords: Cellular automata, memory, topology.

1 THE BEEHIVE CELLULAR AUTOMATON

Cellular Automata (CA) are discrete, spatially explicit extended dynamic systems. A CA system is composed of adjacent cells or sites arranged as a regular lattice, which evolves in discrete time steps. Each cell is characterized by an internal state whose value belongs to a finite set. The updating of these states is made simultaneously according to a common local transition rule involving only the neighborhood of each cell. Thus, if \( \sigma_i^{(T)} \) is taken to denote the value of cell \( i \) at time step \( T \), the site values evolve by iteration of the mapping:

\[
\sigma_i^{(T+1)} = \phi(\sigma_j^{(T)} \in \mathcal{N}_i),
\]

where \( \phi \) is an arbitrary function which specifies the cellular automaton rule operating on the neighborhood \( \mathcal{N} \) of the cell \( i \).

This paper deals with a particular two-dimensional, three-state (0, 1, and 2) rule implemented in the hexagonal tessellation: the beehive rule introduced in [1], [2]. The beehive rule exhibits mobile localized

---

1 The beehive rule is a totalistic rule which assigns the following outputs to each of the 28 possible frequencies of the three states (2, 1, and 0) in the hexagonal tessellation: (0, 0, 6) \( \rightarrow \) 0 (0, 1, 5) \( \rightarrow \) 1 (0, 2, 4) \( \rightarrow \) 2 (0, 3, 3) \( \rightarrow \) 1 (0, 4, 2) \( \rightarrow \) 2 (0, 5, 1) \( \rightarrow \) 0 (0, 6, 0) \( \rightarrow \) 0 (1, 0, 5) \( \rightarrow \) 0 (1, 1, 4) \( \rightarrow \) 2
FIGURE 1
The beehive cellular automaton starting from a single $\sigma = 1$ active cell (light). Darker cells are at state 2. Evolving patterns up to $T = 9$.

FIGURE 2
The beehive cellular automaton starting at random. Evolving patterns up to $T = 12$.

patterns—gliders—which dominate the lattice at the concluding phase of development. Figure 1 shows an example of the beehive rule starting from a single $\sigma = 1$ active cell. This scenario produces six gliders at $T = 9$. Figure 2 shows the evolving patterns of the beehive rule starting at random in a lattice of size $27 \times 27$, in which case only a single glider remains alive at $T = 6$. Periodic boundary conditions were imposed on the edges in Fig. 2, so the glider “reappears” on the left side after reaching the right side. To better visualize this transition, the inactive ($\sigma = 0$) cells have been made apparent in the last four patterns of Fig. 2.

In this article, the effect of geometrically discounted memory implemented on the standard beehive cellular automaton is studied in Section 2. The

$$(1, 2, 3) \rightarrow (2, 1, 3, 2) \rightarrow (2, 1, 4, 1) \rightarrow (1, 1, 5, 0) \rightarrow (1, 2, 0, 4) \rightarrow (0, 2, 1, 3) \rightarrow (2, 2, 2, 1) \rightarrow (2, 2, 3, 1) \rightarrow (2, 2, 3, 1)$$

$$(2, 4, 0) \rightarrow (0, 3, 0, 3) \rightarrow (0, 3, 1, 2) \rightarrow (2, 3, 2, 1) \rightarrow (2, 3, 3, 0) \rightarrow (0, 4, 0, 2) \rightarrow (0, 4, 1, 1) \rightarrow (0, 4, 2, 0) \rightarrow (2, 5, 0, 1) \rightarrow (2, 5, 1, 0) \rightarrow (6, 0, 0) \rightarrow 0.$$
The Beehive Cellular Automaton with Memory

reversible implementation of the bee hive rule with memory is considered in Section 3. Sections 4 and 5 deal with structurally dynamic CA, in which not only cell state values but also links vary dynamically. Section 5 explores the effect of alternative memory models on the bee hive rule.

2 CELLULAR AUTOMATA WITH MEMORY

Standard CA are ahistoric (memoryless): the transition function depends on the neighborhood configuration of the cells only at the preceding time step. Historic memory can be embedded in the CA dynamics by featuring every cell by a mapping of its states in the previous time steps. What is here proposed is to maintain the transition rules \( \phi \) unaltered, but make them act on the cells featured by a function of their previous states:

\[
\sigma_i^{(T+1)} = \phi(s_j^{(T)} \in N_j), \quad s_i^{(T)}
\]

being a state function of the series of states of the cell \( i \) up to time-step \( T \).

Thus, after time-step \( T \) every cell is pre-featured by an un-rounded weighted mean of its previous states:

\[
m_i^{(T)}(\sigma_i^{(1)}, \sigma_i^{(2)}, \ldots, \sigma_i^{(T)}) = \frac{\sigma_i^{(T)} + \sum_{t=1}^{T-1} \alpha^{T-t} \sigma_i^{(t)}}{1 + \sum_{t=1}^{T-1} \alpha^{T-t}} \equiv \frac{\sigma_i^{(T)}}{\Omega(T)} \quad (1)
\]

and then the \( s \in \{0, 1, 2\} \) state values are obtained by comparing the \( m \) ones to the hallmarks 1/2 and 3/2, assigning the last state value in the case of an equality to any hallmark:

\[
s_i^{(T)} = \begin{cases} 
0 & \text{if } m_i^{(T)} < 0.5 \\
\sigma_i^{(T)} & \text{if } m_i^{(T)} = 0.5 \\
1 & \text{if } 0.5 < m_i^{(T)} < 1.5 \\
\sigma_i^{(T)} & \text{if } m_i^{(T)} = 1.5 \\
2 & \text{if } m_i^{(T)} > 1.5
\end{cases} \quad (2)
\]

Memory becomes operative after \( T = 2 \), with the initial assignation \( s_i^{(1)} = \sigma_i^{(1)} \).

In the three-state scenario, geometrically discounted memory does not affect the scenario if \( \alpha \leq 0.25 \). In CA with \( k \) states, the limit effective value for \( \alpha \) is \( \alpha = 1/2(k-1) \) [8].

Figure 3 shows the effect of memory on the bee hive rule considering two values of the memory factor \( \alpha \): the small value 0.3 and the maximum
FIGURE 3
Effect of memory on the beehive cellular automaton starting as in Fig. 1. For every value of the memory factor $\alpha$, the first row of patterns shows the evolution of $\sigma$, and the second row that of $s$.

$\alpha = 1.0$, the case of full memory in which one:

$$m_i^{(T)} = \frac{1}{T} \sum_{t=1}^{T} \sigma_i^{(t)}.$$  

In the latter case, memory affects the evolution as early as after $T = 3$: the outer $\sigma = 1$ cells at $T = 3$, with state history 001, are featured as dead, and the $\sigma = 2$ cells at $T = 3$, with state histories 002 or 012, are featured as $\sigma = 1$ cells. Consequently, the pattern of the ahistoric and full memory models diverge from $T = 4$ in Fig. 3, leading to a series of patterns which do not progress far away from the initial seed in the full memory model. The rounding mechanism [2] is rather biased toward the state 1, which tends to generate $s = 1$ featuring states, as occurs in the full memory model in Fig. 3. With memory of level $\alpha = 0.3$, only the six outer $\sigma = 2$ cells at $T = 3$, with history 002, are not featured by their last state but as $s = 1$. But this fact determines also a dramatic change in the evolution, featured by structured beehive-like patterns, unlike the ahistoric model.

Figure 4 shows the effect of memory on the beehive rule starting at random as in Fig. 2. At variance with what happens with no memory, the dynamics is much less dramatic, so no gliders but a fairly regular distribution of states remains. Figure 5 shows the evolution of the densities of 0, 1 and 2 state values in the scenario of Fig. 4. In these extreme memory scenarios, the density of state 1 is the lowest one, as induced by the beehive rule which assigns 1 state value only in four of the twenty eight possible frequencies (see footnote 1).
The effect on CA of memory in cells has been studied in the references by Alonso-Sanz et al., in the two-state scenario and in the one dimensional three-state one. As a general rule, geometrically discounted memory notably alters the CA dynamics. Not only at high values of the memory factor $\alpha$: the effect is also dramatic at low $\alpha$ levels as the aspect of the patterns with memory differs notably from that of the ahistoric ones.

Note that the memory mechanism here adopted is *accumulative* in its demand of knowledge of past history: to calculate the memory charge $\omega_i^{(T)}$ stated in [1], it is not necessary to know the whole $\{\sigma_i^{(t)}\}$ series, while it suffices to proceed sequentially as:

$$\omega_i^{(T)} = \alpha \omega_i^{(T-1)} + \sigma_i^{(T)}.$$

In order to work with integers and save computing demands, instead of comparing the $m$ values to the hallmarks $1/2$ and $3/2$, it is preferable to compare the $2\omega$ values to the hallmarks 1 and 3 [8].
Let us point out here that the implementation of memory adopted in this work, keeping the transition rule unaltered but applying it to a function of previous states, can be adopted in any dynamical system. In an earlier work [5]–[7] we explored the effect of embedding this kind of memory into discrete dynamical systems: \( x_{T+1} = f(x_T) \) by means of \( x_{T+1} = f(m_T) \) with \( m_T \) being a mean value of past states. We have studied this approach in what is perhaps the canonical example: the logistic map, which becomes with memory \( x_{T+1} = m_T + \lambda m_T (1 - m_T) \). In [4], we studied the effect of memory in a particular Markovian stochastic process (the random walk), \( p_{T+1} = p_T M \) by means of \( p_{T+1} = \pi_T M \), with \( \pi_T \) being a weighted mean of the probability distributions up to \( T \). Returning to the CA scenario, memory can be embedded in continuous CA (or Coupled Map Lattices), in which case the state variable ranges in \( \mathbb{R} \), so \( \sigma_i(T+1) = \psi(\sigma_i(T) \in \mathcal{N}_i(T)) \), where \( \psi \) is a continuous function. The formulation with memory will be:

\[
\sigma_i(T+1) = \psi(\mathcal{M}_i(T) \in \mathcal{N}_i(T)),
\]

as illustrated in a one-dimensional example in [12].

### 3 REVERSIBLE CELLULAR AUTOMATA WITH MEMORY

It should be emphasized that the memory mechanism considered here is different from that of other CA with memory reported in the literature. Typically, higher-order-in-time rules incorporate memory into the transition rule. Thus, in second order in time rules, the transition rule operates as:

\[
\sigma_i(T+1) = \Phi(\sigma_j(T) \in \mathcal{N}_i, \sigma_j(T-1) \in \mathcal{N}_j).
\]

Particularly interesting is the reversible formulation based on the subtraction modulo \( k = 3 \) (noted \( \ominus \)):

\[
\sigma_i(T+1) = \phi(\sigma_j(T) \in \mathcal{N}_i) \ominus \sigma_j(T-1),
\]

reversed as

\[
\sigma_i(T-1) = \phi(\sigma_j(T) \in \mathcal{N}_i) \ominus \sigma_j(T+1).
\]

Figure 6 shows the evolving patterns of the reversible formulation starting as in Fig. 1. As a rule, the pattern at \( T = 0 \) in the reversible simulations

![FIGURE 6](image)

Reversible formulation of the beehive rule starting as in Fig. 1.
FIGURE 7
Effect of memory in the reversible formulation of the beehive rule stated in Fig. 6.

here is the same as that at $T = 1$. Thus, the central cell at $T = 2$ in Fig. 6 becomes at state 2 (red): $0 \oplus 1 = 2$.
To preserve reversibility, the reversible formulation with memory must be:

$$\sigma_i^{(T+1)} = \phi(s_j^{(T)} \in N_i) \odot \sigma_i^{(T-1)} [9].$$

The general considerations regarding the inertial effect of memory in the irreversible scenario apply in the reversible implementation. So, starting as in Fig. 6 but with full memory, the disruption induced by a single $\sigma = 1$ cell is much restrained to the proximity of the site where it appeared, as shown in Figure 7.

For reversing from $T$ it is necessary to know not only $\sigma_i^{(T)}$ and $\sigma_i^{(T+1)}$ but also $\omega_i^{(T)}$ to be compared to $\Omega(T)$, to obtain:

$$s_i^{(T)} = \begin{cases} 
0 & \text{if } 2\omega_i^{(T)} < \Omega(T) \\
\sigma_i^{(T+1)} & \text{if } 2\omega_i^{(T)} = \Omega(T) \\
1 & \text{if } \Omega(T) < 2\omega_i^{(T)} < 3\Omega(T) \\
\sigma_i^{(T+1)} & \text{if } 2\omega_i^{(T)} = 3\Omega(T) \\
2 & \text{if } 2\omega_i^{(T)} > 3\Omega(T). 
\end{cases}$$

Then to obtain $s_i^{(T-1)}$, it is necessary to obtain:

$$\omega_i^{(T-1)} = \frac{1}{\alpha}(\omega_i^{(T)} - \sigma_i^{(T)}).$$
FIGURE 8
The reversible beehive cellular automaton starting at random as in Fig. 2. Evolution up to $T = 4$.

But in order to avoid the division by the memory factor (recall that operations with real numbers are not exact in computer arithmetic), it is preferable to work with $\gamma(T-1) = \omega(T) - \sigma(T)$, and to compare these values to $\Gamma(T-1) = \sum_{t=1}^{T-1} \alpha^{T-t}$. This leads to:

$$s^{(T-1)}_i = \begin{cases} 0 & \text{if } 2\gamma^{(T-1)}_i < \Gamma(T-1) \\ \sigma_i^{(T)} & \text{if } 2\gamma^{(T-1)}_i = \Gamma(T-1) \\ 1 & \text{if } \Gamma(T-1) < 2\gamma^{(T-1)}_i < 3\Gamma(T-1) \\ \sigma_i^{(T)} & \text{if } 2\gamma^{(T-1)}_i = 3\Gamma(T-1) \\ 2 & \text{if } 2\gamma^{(T-1)}_i > 3\Gamma(T-1). \end{cases}$$

Continuing in the reversing process:

$$\gamma^{(T-2)}_i = \gamma^{(T-1)}_i - \alpha \sigma^{(T-1)}_i$$

and $\Gamma(T-2) = \sum_{t=1}^{T-2} \alpha^{T-t}$. In general:

$$\gamma^{(T-\tau)}_i = \gamma^{(T-\tau+1)}_i - \alpha^{T-\tau} \sigma^{(T-\tau+1)}_i$$

and $\Gamma(T-\tau) = \sum_{t=1}^{T-\tau} \alpha^{T-t}$.

$$s^{(T-\tau)}_i = \begin{cases} 0 & \text{if } 2\gamma^{(T-\tau)}_i < \Gamma(T-\tau) \\ \sigma_i^{(T-\tau+1)} & \text{if } 2\gamma^{(T-\tau)}_i = \Gamma(T-\tau) \\ 1 & \text{if } \Gamma(T-\tau) < 2\gamma^{(T-\tau)}_i < 3\Gamma(T-\tau) \\ \sigma_i^{(T-\tau+1)} & \text{if } 2\gamma^{(T-\tau)}_i = 3\Gamma(T-\tau) \\ 2 & \text{if } 2\gamma^{(T-\tau)}_i > 3\Gamma(T-\tau). \end{cases}$$

Figure 8 shows the effect of memory on the reversible implementation of the beehive rule starting at random as in Fig. 2. At variance with what happens in the non-reversible formulation, the dynamics in Fig. 8 does not lead to gliders or other isolated structures. Figure 9 shows the evolution of the densities of 0, 1 and 2 state values in the scenario of Fig. 8.
densities vary in a narrow range, not only in the ahistoric scenario but also with memory, as shown in Fig. 9.

4 REVERSIBLE STRUCTURALLY DYNAMIC CELLULAR AUTOMATA

Structurally dynamic cellular automata (SDCA) were suggested by Ilachinski and Halpern [14], [15]. The essential new feature of this model is that the connections between the cells are allowed to change according to rules similar in nature to the state transition rules associated with the conventional CA. This means that given certain conditions, specified by the link transition rules, links between rules may be created and destroyed; the neighborhood of each cell is now dynamic rather than fixed throughout the automaton, so state and link configurations of an SDCA are both dynamic and are continually interacting.

In the Ilachinski and Halpern model, an SDCA consists of a finite set of binary-valued cells numbered 1 to $N$ whose connectivity is specified by an $N \times N$ connectivity matrix in which $\lambda_{ij}$ equals 1 if cells $i$ and $j$ are connected; 0 otherwise. So, now:

$$\mathcal{N}^T_i = \{ j/\lambda^T_{ij} = 1 \}$$

and

$$\sigma^{(T+1)}_i = \phi(\sigma^{(T)}_j \in \mathcal{N}^T_i).$$

The (geodesic) distance between two cells $i$ and $j$, $\delta_{ij}$, is defined as the number of links in the shortest path between $i$ and $j$. We say that $i$ and $j$ are direct neighbors if $\delta_{ij} \leq 1$, and that $i$ and $j$ are next-nearest neighbors if $\delta_{ij} = 2$. There are two types of link transition functions in an SDCA:
FIGURE 10
The ahistoric structurally dynamic beehive rule described in Section 4 starting from a ring of \( \sigma = 1 \) cells.

couplers and decouplers, the former add new links, the latter remove links. The set of coupler and decoupler determines the link transition rule

\[
\lambda_{ij}^{(T+1)} = \psi \left( \lambda_{ij}^{(T)}, \sigma_i^{(T)}, \sigma_j^{(T)} \right).
\]

Instead of introducing the formalism of the SDCA, we deal here with just one example in which the decoupler rule removes all links connected to cells in which neither value is zero (\( \lambda_{ij}^{(T)} = 1 \rightarrow \lambda_{ij}^{(T+1)} = 0 \) iff \( \sigma_i^{(T)} + \sigma_j^{(T)} = 0 \)) and the coupler rule adds links between all next-nearest neighbor sites in which both values are not dead (\( \lambda_{ij}^{(T)} = 0 \rightarrow \lambda_{ij}^{(T+1)} = 1 \) iff \( \sigma_i^{(T)} > 0, \sigma_j^{(T)} > 0 \) and \( \delta_{ij}^{(T)} = 2 \)). All of these totalistic rules are applied at the same time (in parallel).

Let us consider the case of Figure 10, in which, the hexagonal tessellation\(^2\) is initially seeded with a ring of \( \sigma = 1 \) cells (as at \( T = 2 \) in Fig. 1). After the first iteration, at time-step \( T = 2 \), most of the lattice structure has decayed as an effect of the decoupler rule, so that the active value cells and links are confined into a small region. Evolution in Fig. 10 is shown up to \( T = 4 \), as from this time-step the pattern remains unaltered. By varying the wiring, the frequency of states is not obliged to add up to six. But the beehive rule has not been altered, so if the sum of frequencies is over six, the cell value remains unaltered. This explains what happens in Fig. 10, where the evolution of a typical non-dead cell at \( T = 1 \) is:

The Fredkin’s reversible construction is feasible in the SDCA scenario, extending the \( \ominus \) operation also to links:

\[
\lambda_{ij}^{(T+1)} = \psi \left( \lambda_{ij}^{(T)}, \sigma_i^{(T)}, \sigma_j^{(T)} \right) \ominus \lambda_{ij}^{(T-1)}.
\]

\(^2\) With next-nearest neighborhood.
The Beehive Cellular Automaton with Memory

Figure 11 shows the evolution of the reversible formulation of the SDCA of Fig. 10 up to \( T = 5 \). At variance with what happens in the irreversible formulation in Fig. 10, the initial lattice structure does not decay at \( T = 2 \) (nor at posterior time-steps) because of the subtraction of the structure at \( T = 0 \) (at \( T - 1 \)), supposed to be the same as that at \( T = 1 \). Link transition rules do not alter auto-connections, but subtraction of patterns may do so. Thus, for example, in Fig. 11 every cell is auto-connected at \( T = 0 \) and \( T = 1 \), but the subtraction of these patterns leads to the complete disappearance of auto-connections at \( T = 2 \). Auto-connections are not represented in figures, but of course they affect the mass updating. To ease the visualization, the wiring of border cells is not represented.

5 A REVERSIBLE STRUCTURALLY DYNAMIC CELLULAR AUTOMATON WITH MEMORY

Memory can be embedded in links in a manner similar to that in state values, so the link between any two cells is featured by a mapping of its previous values:

\[
l_{ij}^{(T)} = \text{round}(m_{ij}^{(T)}), \quad \lambda_{ij}^{(T)} = \lambda_{ij}^{(T-1)} + \sum_{t=1}^{T-1} \alpha \lambda_{ij}^{(T-t)} = \omega_{ij}^{(T)} + \alpha \lambda_{ij}^{(T)}.
\]

with

\[
\omega_{ij}^{(T)} = \lambda_{ij}^{(T)} + \sum_{t=1}^{T-1} \alpha \lambda_{ij}^{(T-t)} = \alpha \lambda_{ij}^{(T)}.
\]

The distance between two cells in a historic model \( (d_{ij}) \) is defined in terms of the \( \lambda \) instead of the \( \omega \) values, so that \( i \) and \( j \) are direct neighbors if \( d_{ij} = 1 \), and are next-nearest neighbors if \( d_{ij} = 2 \); \( N_{ij}^{(T)} = \{ j/d_{ij}^{(T)} = 1 \} \). Generalizing the approach to embedded memory introduced in Section 2, the unchanged transition rules (\( \phi \) and \( \psi \)) may operate on the featured link.
and mass values:
\[
\sigma_i^{(T+1)} = \phi \left( s_j^{(T)} \in N_i \right), \quad \lambda_{ij}^{(T+1)} = \psi \left( l_{ij}^{(T)}, s_i^{(T)}, s_j^{(T)} \right) [3].
\]

A generalization of the Fredkin’s reversible construction is feasible in the SDCA scenario endowed with memory as:
\[
\sigma_i^{(T+1)} = \phi \left( s_j^{(T)} \in N_i^{(T)} \right) \ominus \sigma_i^{(T-1)}, \quad \lambda_{ij}^{(T+1)} = \psi \left( l_{ij}^{(T)}, s_i^{(T)}, s_j^{(T)} \right) \ominus \lambda_{ij}^{(T-1)}.
\]

Now, for reversing from \( T \) it is necessary to know not only \( \sigma_i^{(T)}, l_{ij}^{(T)}, \sigma_i^{(T+1)}, \) and \( l_{ij}^{(T+1)} \), but also \( \omega_i^{(T)} \) and \( \omega_{ij}^{(T)} \), proceeding for reversing in connections as stated for mass values in Section 3.

Figure 12 shows the initial effect of full memory (\( \alpha = 1.0 \)) in the initial scenario of Fig. 11.

Figure 13 shows the patterns at \( T = 13 \) for the ahistoric and full memory reversible SDCA with the initial steps shown in Figs. 11 and 12. In the ahistoric model, the web of connections is so dense in its central area that it is impossible to discern it. The web appears dramatically cleared in the historic model with full memory. The clearing of the web of connections together with a restraint in the advance of mass, mark the inertial effect of memory.

FIGURE 12
The reversible structurally dynamic beehive rule with full memory, starting as in Fig. 11. Evolution from \( T = 2 \) up to \( T = 5 \).

FIGURE 13
The structurally dynamic reversible beehive rule at \( T = 13 \). Ahistoric (left) and full memory models, starting as in Fig. 11.
FIGURE 14
The evolving patterns of the **beehive** rule, with integer-based memory $\delta(t) = t$, starting as in Fig. 1.

6 OTHER MEMORIES

Average-like memory models can readily be proposed by generalizing the memory charges as:

$$
\omega_i^{(T)} = \sum_{t=1}^{T} \delta(t) \sigma_i^{(t)} , \quad \omega_{ij}^{(T)} = \sum_{t=1}^{T} \delta(t) \lambda_{ij}^{(t)}, \quad \text{with} \quad \Omega(T) = \sum_{t=1}^{T} \delta(t) .
$$

The geometric discount model considered till now ($\delta(t) = \alpha^{T-t}$) is just one of the many possible weighting functions.

Alternatively, previous states can be pondered with the weight: $\delta(t) = t^c$ [11]. Choosing integer $c$ parameter values allows working only with integers by comparing the $2\omega_i$ and $2\omega_{ij}$ figures across the lattice to the factor $\Omega(T)$. For $c = 0$, we have the full historic model; for $c = 1$ it is $\delta(t) = t$ with $\Omega(T) = T(T + 1)/2$. The larger the value of $c$, the more heavily the recent past is taken into account, and consequently the closer the scenario to the ahistoric one. Figure 14 shows an example of the effect of $\delta(t) = t$ integer-based memory implementation.

Another weight with the same integer-based property, is $c^t$ (again provided that $c$ is an integer). This memory weight is not operative with cells and links with two states, but it becomes operative when allowing three states. Figure 15 shows the effect of $\delta(t) = 2^t$ memory starting as in Fig. 1.

FIGURE 15
The evolving patterns of the **beehive** rule, with integer-based memory $\delta(t) = 2^t$, starting as in Fig. 1.
FIGURE 16
The evolving patterns of the reversible formulation of the beehive rule, with integer-based memory weights, starting as in Fig. 1.

Figure 16 shows evolving patterns of the reversible implementation with the integer-based memories in the scenarios of Figs. 14 and 15. Reversing is easier in the integer-based memory scenarios than in that of geometric discount as $\omega_j^{(T)} = \omega_j^{(T-1)} + \delta(T)\sigma_j^{(T)}$ and $\omega_{ij}^{(T)} = \omega_{ij}^{(T-1)} + \delta(T)\lambda_{ij}^{(T)}$ readily reverses integer-based, without the computational inconvenience of division by $\alpha$. Working only with integers (à la CA) is a clear computational advantage. Nevertheless, the integer-based weights, $c^e$ and $t^e$ share the same drawback: they explode, at high values of $t$, even for $c = 2$.

The trailing memory can also be limited to as much as the three last time steps, in which case memory can be implemented by means of one-dimensional CA rules:

$$s_i^{(T)} = \phi(\sigma_i^{(T-2)} + \sigma_i^{(T-1)} + \sigma_i^{(T)}), \quad l_{ij}^{(T)} = \phi(\lambda_{ij}^{(T-2)} + \lambda_{ij}^{(T-1)} + \lambda_{ij}^{(T)})$$

We consider here only totalistic rules: rules in which the value of a site depends only on the sum of the values of the neighbors, and not on their individual values, so $s_i^{(T)} = \phi(\sigma_i^{(T-2)} + \sigma_i^{(T-1)} + \sigma_i^{(T)})$. Following Wolfram’s notation, $k = 3$ rules are characterized by a sequence of ternary values $(\beta_s)$ associated with each of the seven possible values of the sum $(s)$ of the neighbors:

$$(\beta_6, \beta_5, \beta_4, \beta_3, \beta_2, \beta_1, \beta_0)_{\text{ternary}} \equiv \sum_{i=0}^{6} \beta_i 3^i \quad \text{decimal} = \mathcal{R}$$

The rules are conveniently specified by a decimal integer, to be referred to as their rule number ($\mathcal{R}$) which for $k = 3$ will range from 0 to 2186. In Figure 17, cells are featured by the parity of the last three state values:

$$s_{i}^{(T)} = \sigma_{i}^{(T-2)} + \sigma_{i}^{(T-1)} + \sigma_{i}^{(T)} \mod 2$$

in rule 273 (0101010) and binary number (0101020) in rule 276. In the latter case, the evolution produces the null configuration (all cells dead) at $T = 9$, but this does not mean extinction: at $T = 10$ a new pattern appears (the same that at $T = 3$). This odd cataleptic phenomenon is not feasible with no memory.
FIGURE 17
The evolving patterns of the bee hive rule, with two parity rules of the last three states, starting as in Fig. 1.

FIGURE 18
The evolving patterns of the bee hive rule, with the bee hive as memory rule, starting as in Fig. 1.

Last but not least, in Figure 18 the memory rule is the bee hive rule acting after $T = 6$. We expected gliders, but they do not appear in Fig. 18, not even starting at random.

7 CONCLUSION

Geometrically discounted memory produces an inertial effect that induces the preservation of some of the main features of the initial scenario. This notably alters the ahistoric dynamics, even if a low level of memory is implemented. So, i) starting from a single $\sigma = 1$ active cell, the bee hive rule with memory tends to generate bee hive or snowflake-like structures around the initial seed, and ii) fairly random patterns remain starting at random, although with a small proportion of 1 state values as induced by the bee hive rule itself.
Some critics may argue that memory is not in the realm of CA (or even of Dynamic Systems), but we believe that the subject is worth studying. At least CA with memory can be considered as a promising extension of the basic paradigm. A major impediment to modeling with CA stems from the difficulty of utilizing the CA complex behavior to exhibit a particular behavior or perform a particular function: embedding memory in states (and links) broadens the spectrum of CA as a tool for modeling. It is likely that in some contexts, a transition rule with memory could match the “correct” behavior of the CA system of a given complex system.

Apart from their potential applications, CA with memory have an aesthetic and mathematical interest on their own. The study of the effect of memory on CA has been rather neglected\(^3\). Nevertheless, it seems plausible that further study of the effect of memory on CA (and Lattice Gas Automata) should turn out to be profitable. This encloses not only the geometrically discounted memory but any other type of memory acting in the basic CA paradigm scenario, and in other CA scenarios such as structurally dynamic CA, in which case memory may be embedded also in links, as explored in this paper.

Perhaps, as a result of a further full rigorous study of CA with memory, it will be possible to paraphrase T. Toffoli in presenting CAM -as an alternative to (rather than an approximation of) integral equations in modeling. In particular, to Volterra Integral Equations which appear in the study of many phenomena which are important in applied sciences, such as population dynamics, diffusion, neural networks, and so on.

**ACKNOWLEDGMENT**

This work was made during the elaboration of the Spanish MEC PR-2006-0081 proposal.

**REFERENCES**


\(^3\) Wuensche and Lesser ([16], p. 15), at least, state that CA with memory in cells “would result in a qualitatively different behavior”.


