Nat Comput (2012) 11:277–287
DOI 10.1007/s11047-012-9310-4

# An analysis of different types and effects of asynchronicity in cellular automata update schemes

**Stefania Bandini · Andrea Bonomi · Giuseppe Vizzari**

Published online: 24 February 2012


**Abstract** This paper introduces the problematics deriving from the adoption of asynchronous update schemes in CA models. Several cellular automata update schemes and a tentative classification of such schemes are introduced and discussed. In order to analyze the effects of the different update schemes, a class of simple CA—called One neighbor binary cellular automata (1nCA)—is then introduced. An overview of the general features of 1nCA is described, then the effects of six different updates schemes on all the class of 1nCA are described.

**Keywords** Cellular automata · Asynchronous CA · Asynchronous CA update schemes

## 1 Introduction

A fundamental feature of Cellular Automata models is the fact that time is considered as discrete and state updates occur synchronously and in parallel. Nevertheless, this assumption can be quite restrictive on the class of phenomena that can be modeled and several authors have argued that asynchronous models are viable alternatives to synchronous ones and suggest that asynchronous models should be preferred where there is no evidence of a global clock in the modeled reality (Paolo et al. 2000). Moreover, as argued in Schönfisch and de Roos (1999), sometimes the modeling activity is aimed at achieving a form of "model stability" for which the qualitative results of the model depend only on the basic assumptions derived from the modeled system, and they are not to be ascribed to the actual details of the employed model, such as the update scheme of a CA.

In general, when cells updating does not take place simultaneously we talk about an asynchronous CA; this term thus does not state any precise property of this kind of model, it essentially expresses the fact that a property (i.e., synchronicity in cells updates) is not valid. One of the main aims of this paper is to clarify that there are substantially different ways in which a CA can be asynchronous and to present an investigation of the these different ways and their implications on the overall system dynamics, by defining and adopting a specific sample CA model for this purpose.

One of the first works proposing an asynchronous updating scheme is described in Kanada (1994): the model is characterized by different cell updating schemes, basically sequential ones, in which a single cell is updated at each time step. The order of the updating sequence is defined as one of the following three methods: *Random order*, *Fixed Random Order*, and *Interlaced order*.

In Page (1997), issues and implications of asynchronous update schemes is analyzed from the point of view of modeling economical systems: in this approach, cells that benefit most (according to some utility function) update their state first. This incentive based update approach, however, is clearly more suited to characterize agent–based

The work presented in this paper has been partially funded by the University of Milano-Bicocca within the project "Fondo d'Ateneo per la Ricerca - anno 2010".

S. Bandini · A. Bonomi · G. Vizzari (✉)
Complex Systems and Artificial Intelligence (CSAI) Research Center, Department of Informatics, Systems and Communication (DISCo), University of Milan, Bicocca Viale Sarca 336/14, 20126 Milano, Italy
e-mail: vizzari@disco.unimib.it

S. Bandini
e-mail: bandini@disco.unimib.it

A. Bonomi
e-mail: bonomi@disco.unimib.it

models, since the extrapolation of a "utility from updating" function is much more natural in models comprising self-interested entities as first class abstractions rather than simple cells.

In Cornforth et al. (2005) the adopted notion of agent is instead much more similar to CA cells; three classes of update scheme are identified in this work: *Synchronous Update*, *Random Asynchronous* (RAS), and *Ordered Asynchronous* (OAS). The first scheme is the traditional CA updating scheme; according to the RAS scheme, at any given time individuals to be updated are selected at random according to some probability distribution. In the OAS update process, the updating of individual states follows a systematic pattern. The authors consider a total of six update patterns, including two RAS schemes and three OAS scheme: *Synchronous Scheme*, *Random Independent (RAS)*, *Random Order (RAS)*, *Cyclic (OAS)*, *Clocked (OAS)*, and *Self-Sync*. The author chose to implement local synchronicity by using a coupled oscillator approach. The period of each timer is adjusted after an update so as to more closely match the period of other cells in its neighborhood.

Another relevant study about the effects of asynchronicity in a CA model is described in Fatès et al. (2005): in particular, the authors focus on the robustness of one dimensional asynchronous CAs characterized by a RAS scheme.

In addition to CA models, a relevant area in which issues and implications of updating schemes have been analyzed is related to random boolean networks Darabos et al. (2007), a model in which not only timing but also spatial constraints are relaxed when compared to traditional CA.

Finally, in addition to the above mentioned works taking an agent-based modeling approach, it is worth mentioning the fact that agent activation issues have been considered both from a general situated agents perspective (Bandini et al. 2005), in application situations considering both the specification and implementation of simulation systems (Bandini and Vizzari 2006) and the definition of protocols for distributed systems (Fang et al. 2005).

The aim of this paper is to provide a comprehensive analysis of different asynchronous update schemes and to evaluate the effect of their adoption in a simplified CA model; the paper is organized as follows: the following section formally introduces a comprehensive set of relevant updating schemes, while Sect. 3 introduces One neighbor binary cellular automata (1nCA), the simple model in which the different update schemes will be exhaustively tested. Section 4 presents a classification of all the possible update rules that can be adopted in the 1nCA model. Section 5 describes the effects of the adoption of the different update schemes for this model, while conclusions and future developments end the paper.

## 2 A classification of update schemes

In order to classify the update schemes, we define the following parameters:

- $p_i^{(t)}$ determines the period between two successive updates of the cell $i$ at the time step $t$, i.e. how many time steps the cell $i$ will wait in order to be updated. The value of $p$ can change during the time, e.g., in the Self-Sync update scheme.
- $l_i^{(t)}$ determines the length of the updating (in terms of time steps) of the cell $i$ at the time step $t$, i.e., after how many time steps the neighbor cells taking into account the new state during their update.
- $d_i$, determines the delay (in terms of time step) before the first update.
- $U^{(t)}$ is the set of cells beginning the update process at the at the time step $t$ simultaneously.
- $u^{(t)} = |U^{(t)}|$ is the number of cells starting the update process at the time step $t$.

Given the above parameter, whose meaning is exemplified in Fig. 1, a set of relevant update schemes will now be presented and discussed. For each update scheme, we give a formal definition that are successively employed for the classification of the update schemes.

### 2.1 Relevant update schemes

#### 2.1.1 Synchronous scheme

All individuals are updated in parallel at each time step; the updating of a cell takes 1 time step. Formally, this update scheme can be characterized as follows:

$$\forall t \in \mathbb{Z}, t > 0, \forall i \in \mathbb{Z}, \leq i < N:$$
$$p_i^{(t)} = 1,\ l_i^{(t)} = 1,\ d_i = 0,\ u^{(t)} = N$$

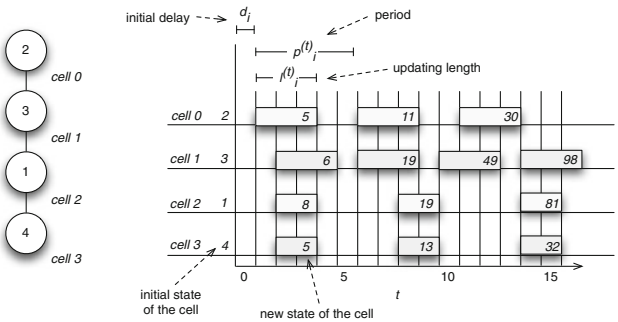


**Fig. 1** A schematic representation of the parameters adopted to describe the update schemes: *left* a sample one dimensional CA composed of four cells is shown, *right* a sample update diagram is used to clarify the meaning of the parameters adopted to describe the update schemes

### 2.1.2 Random independent

At each time step, one and only one cell, chosen at random, is updated; the updating of a cell takes 1 time step. A formal characterization of this update scheme is the following:

$$\forall\, t \in \mathbb{Z}, t > 0,\ \forall\, i \in \mathbb{Z},\ 0 \leq i < N:$$
$$l_i^{(t)} = 1,\ u^{(t)} = 1,\ \exists\, t, i : p_i^{(t)} > 1$$

An example of Random Independent update scheme, related to a 1D CA composed of four cells, is shown in Fig. 2.

### 2.1.3 Random order

All nodes are updated in random order. After the updating of all the nodes, the order is changed. The updating of each cell takes 1 time step. The maximum length of the update period is less than $2N$. This update scheme can be formally described as follows:

$$\forall\, t \in \mathbb{Z}, t > 0, \forall\, i \in \mathbb{Z},\ 0 \leq i < N:$$
$$p_i^{(t)} < 2N,\ l_i^{(t)} = 1,\ d_i < N,\ u^{(t)} = 1$$

An update interval $[\alpha, \omega]$ so that $\forall z \in \mathbb{Z}, z > 0,\ \alpha = 1 + zN,\ \omega = (z+1)N$ can be defined. In every update interval, each cell is update exactly once:

$$\forall\, i \in \mathbb{Z}, 0 \leq i < N,$$
$$\forall\, t_n \in \mathbb{Z}, \alpha \leq t_n \leq \omega,\ \forall\, t_m \in \mathbb{Z}, \alpha \leq t_m \leq \omega,$$
$$c_i \in U^{(t_n)},\ c_i \in U^{(t_m)} \iff t_n = t_m$$

An example of Random Order update scheme, related to a 1D CA composed of four cells, is shown in Fig. 3.

### 2.1.4 Cyclic

At each time step a node is chosen according to a fixed update order. The update scheme can be formally described as follows:

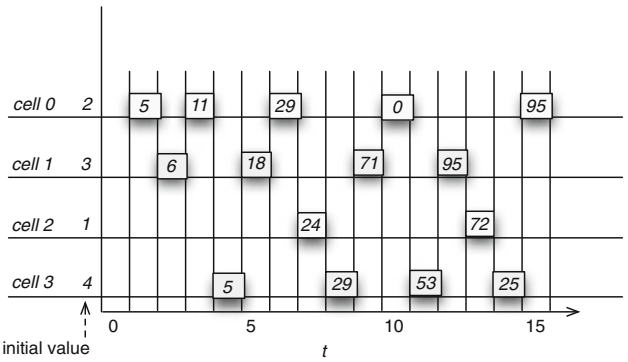


**Fig. 2** An example of Random Independent update scheme, related to a 1D CA composed of four cells

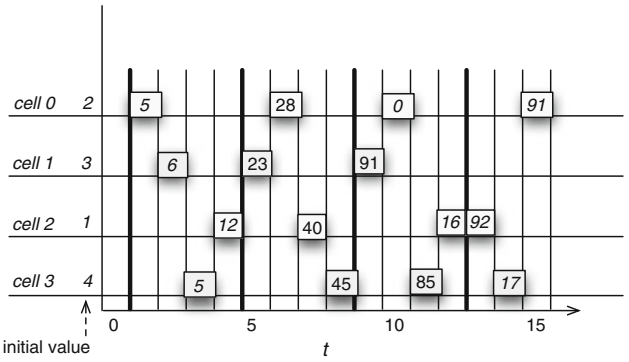


**Fig. 3** An example of Random Order update scheme, related to a 1D CA composed of four cells

$$\forall\, t \in \mathbb{Z}, t > 0, \forall\, i \in \mathbb{Z},\ 0 \leq i < N:$$
$$p_i^{(t)} = N,\ l_i^{(t)} = 1,\ d_i < N,\ u^{(t)} = 1$$

We can identify three subtypes of this update scheme:

- Random cyclic: the update order is decided at random during initialisation of the automaton. This update scheme correspond to the Kannada's *Fixed Random* (Kanada 1994) and Cornforth's *Cyclic OAS* (Cornforth et al. 2005).
- Fixed cyclic-sequential ordered: The update order is fixed in the automaton definition. The cells are updated one-by-one according to their natural order:
  $$d_i = 1 + i;\ q^t = (t-1)\ mod\ N;\ u^{(t)} = \{c_{q^t}\}.$$
- Fixed cyclic-interlaced cyclic: also called *Interlaced Order* in Kanada (1994). The set of cell $u^{(t)}$ to be update at time step $t$ is calculated as $q^t = C(t-1)\ mod\ N$; $u^{(t)} = \{c_{q^t}\}$, where $C$ is a parameter prime to $N$.

An example of Fixed Cyclic-Sequential Ordered update scheme, related to a 1D CA composed of four cells, is shown in Fig. 4.

### 2.1.5 Generic cyclic

It is a generalization of the cyclic update scheme, obtained relaxing the constraint on the updating length. In this update scheme, the updating length is limited only by the period. Formally, this update scheme can be described as follows:

$$\forall\, t \in \mathbb{Z},\ t > 0,\ \forall\, i \in \mathbb{Z},\ 0 \leq i < N:$$
$$p_i^{(t)} = N,\ l_i^{(t)} \leq p^{(t)},\ d_i < N,\ u^{(t)} = 1$$

### 2.1.6 Clocked

A *timer* is assigned to each cell, so that updating is autonomous and proceeds at different rates for different

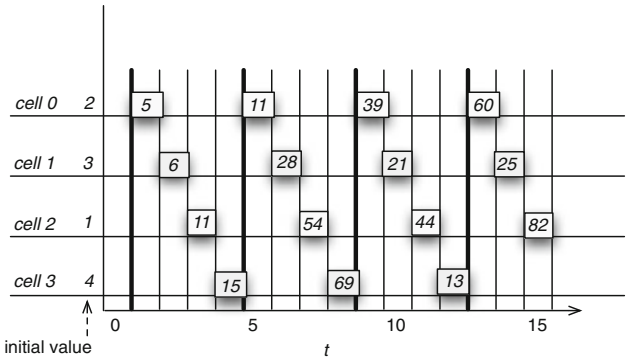


**Fig. 4** An example of Fixed Cyclic-Sequential Ordered update scheme, related to a 1D CA composed of four cells

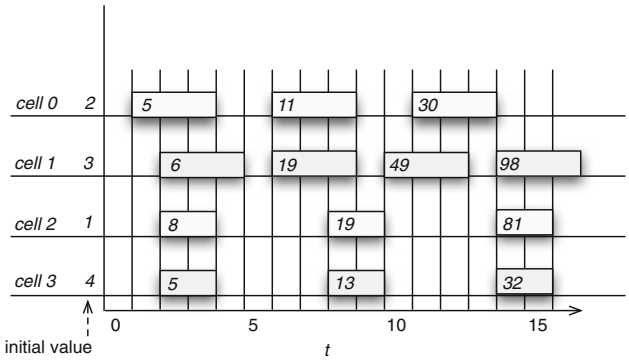


**Fig. 5** An example of Variable Clocked update scheme, related to a 1D CA composed of four cells

cells. The update frequency of each cells is fixed. The scheme can be formally described as follows:

$$\forall t \in \mathbb{Z},\ t > 0,\ \forall i \in \mathbb{Z},\ 0 \leq i < N:$$
$$p_i^{(t)} = p_i^{(0)},\ l_i^{(t)} \leq p_i^{(0)},\ d_i \leq p_i^{(0)}$$

As subtype of the Clocked update scheme is the *Equal Frequency Clocked*. According to this update scheme, every cells has the same update frequency:

$$\forall t \in \mathbb{Z},\ t > 0,\ \forall i \in \mathbb{Z},\ 0 \leq i < N : p_i^{(t)} = p_0^{(0)}$$

#### 2.1.7 Generic clocked

It is a generalization of the cyclic update scheme, obtained relaxing the constraint on the fixed update frequency. The two subtypes of this update scheme are the *Clocked* and *Variable Clocked*. According to the Variable Clocked scheme, a timer is assigned to each cell, so that updating is autonomous and proceeds at different rates for different cells. The updating frequency is not fixed: $\exists t, i : p_i^{(t)} \neq p_i^{(0)}$. The *Self-Sync* update scheme is an example of Variable Clocked scheme. An example of Variable Clocked update scheme, related to a 1D CA composed of four cells is shown in Fig. 5.

### 2.2 Cellular automata update schemes ontology

In this section we present a tentative classification of the previously presented update schemes. In order to manage the complexity deriving from the classification of the schemes according to different features (e.g. the number of cells updated at each time step, the maximum length of the update period) we defined a CA Update Scheme Ontology, a formal conceptualization expressed using the OWL 2 DL[1] language, a W3C endorsed format that can be adopted to define ontologies in terms of classes, properties, instances. This language allows defining relatively rich semantics for the above entities and it is able to represent and manage features like equality and inequality of instances, restrictions (not only related to domain and range, but also to cardinality) and other characteristics of properties, enumerated classes and other formal properties. We defined specific datatype properties to characterize each update scheme (e.g. *hasClock*, *hasUpdateOrder*, *hasFixedPeriod*, *updatedCellsPerTimeStep*); each scheme is characterized by a specific configuration of values for these properties.

We computed the class hierarchy using the Pellet[2] semantic reasoner, an open source Java reasoner for OWL 2 DL. A reasoner is a software able to infer logical consequences from a set of asserted axioms. An example of inference for an OWL DL reasoner is the computation of the class hierarchy. In other words, starting from the update schemes defined as OWL classes, the reasoner is able to inter how update scheme are related by subclass relationships. The result of this automatic classification activity is shown in Fig. 6; in particular, the diagram shows that several update schemes are subclasses of different other classes.

For instance, the traditional Synchronous update scheme is both an instance of the MultiCellUpdate scheme class and also an instance of the class of schemes characterized by the fact that cell update takes exactly one time step (*UpdatingLength1*). Analogously, *RandomIndependent* and *RandomOrder* are instances of the *RandomUpdateScheme* class, being characterized by a random component, but they are also instances of the Sequential update class because they update only one cell per time step.

[1] http://www.w3.org/TR/owl2-primer/.

[2] http://clarkparsia.com/pellet.

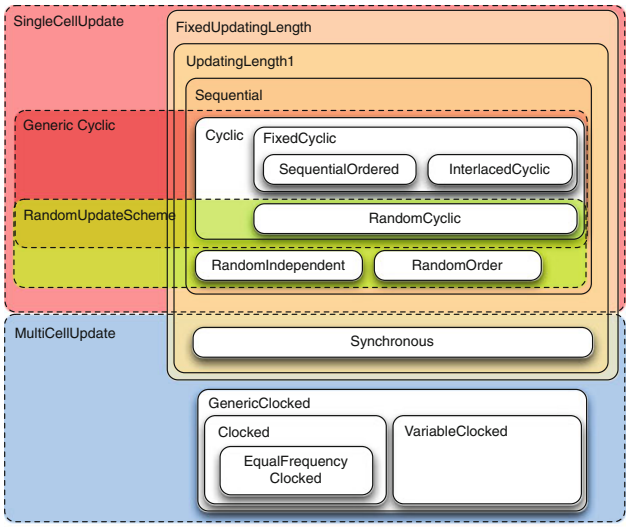


**Fig. 6** A schematic representation of the classes of the CA Update Scheme ontology. The classes *UpdateScheme* and *Asynchronous are not shown for simplicity*

## 3 One neighbor binary cellular automata

One neighbor binary cellular automata (1nCA) is a one-dimensional Cellular Automata, with two possible states per cell. Each cell has two neighbors, left and right, defined to be the adjacent cells on either side, but the update rule consider only one neighbor per step. The neighborhood includes the cell itself and the left or the right adjacent cell and alternates between these two situations at even and odd time steps.

The size of the neighborhood is always 2, so there are four possible patterns for the neighborhood and only 16 possible rules. The number of possible rules is small compared to the 256 possible rules of the Elementary Cellular Automata, so it is easier to exhaustively study the dynamic behavior of the all rules.

These 16 1nCA rules will be referred using the Wolfram notation, with the rule numbers followed by the $\Delta$ symbol to avoid confusion with the Elementary Cellular Automata rules (e.g. "Rule 10" is an Elementary Cellular Automata rule, "Rule 10$\Delta$" is an 1nCA rule).

We call 1nCA the cellular automata $(\mathcal{L}, \mathcal{S}, \mathcal{N}, \mathcal{F})$ where

- $\mathcal{L} = [c_0, c_1, \ldots, c_n]$ is an array of $n$ cells,
- $\mathcal{S} = \{0, 1\}$ is the set of states ($k = 2$),
- $\mathcal{N}_c$ is neighborhood of the cell $c$ and $\forall c : \mathcal{L}\, |\mathcal{N}_c| = 2$,
- $f : \mathcal{S}^2 \rightarrow \mathcal{S}$ is a transition function.

Denoting the cell $c$ at position $i$ as $c_i$, the neighborhood $\mathcal{N}_{c_i}^{(t)}$ of the cell $c_i$ at time $t$ is defined as $\mathcal{N}_{c_i}^{(t)} = [c_i, n_{c_i}^{(t)}]$ where $n_{c_i}^{(t)}$ is the neighbor of the cell, given by

$$n_{c_i}^{(t)} = \begin{cases} c_{i+1} & \text{if } t \text{ is even} \\ c_{i-1} & \text{otherwise} \end{cases}$$

The neighborhood of the cell $c_i$ at different time steps is shown in Fig. 7.

Following the Wolfram's notation, the rules are characterized by a sequence of binary values ($\beta_i \in S$) associated with each of the four possible patterns for the neighborhood. The transition function is defined as:

$$f(c_i, n_{c_i}^{(t)}) = \begin{cases} \beta_0 & \text{if } c_i = 0, n_{c_i}^{(t)} = 0 \\ \beta_1 & \text{if } c_i = 0, n_{c_i}^{(t)} = 1 \\ \beta_2 & \text{if } c_i = 1, n_{c_i}^{(t)} = 0 \\ \beta_3 & \text{if } c_i = 1, n_{c_i}^{(t)} = 1 \end{cases}$$

As shown in Fig. 8, there are 16 possible transition functions, identified by a rule number $R = \sum_{i=0}^{3} \beta_i 2^i$.

The configuration of a cellular automata is a mapping $q : \mathcal{L} \rightarrow \mathcal{S}$ which assigns to each cell of the array $\mathcal{L}$ a state from $\mathcal{S}$. We denoted with $q_t$ the configuration of a cellular automata at time $t$; in particular $q_t = [s_0, s_1, \ldots, s_n] \in \mathcal{S}^n$ (16) where $n$ is the number of cells of $\mathcal{L}$. Given an initial configuration $q_0$, the evolution of an automaton is represented by a sequence of configurations $q_0 \rightarrow q_1 \rightarrow$

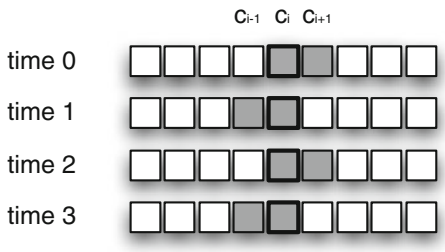


**Fig. 7** The neighborhood of the cell $c_i$ at different time steps

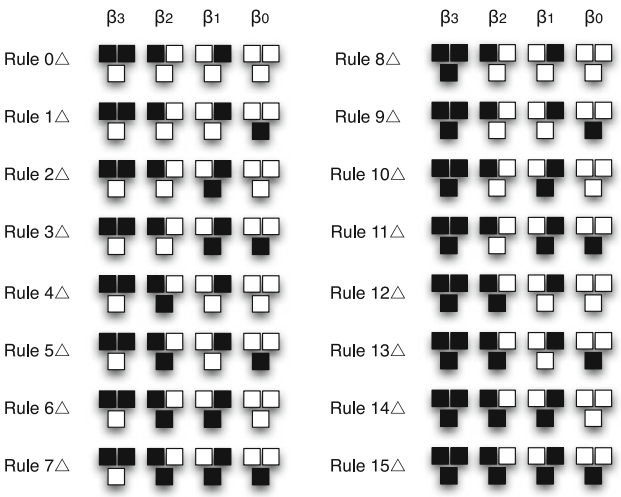


**Fig. 8** Representation of the 16 1nCA transition rules

$q_2 \rightarrow \cdots \rightarrow q_t$. A deterministic finite cellular automaton eventually falls into a cycle (with period $p > 1$) or a fixed point ($p = 1$):

$$q_t \rightarrow q_{t+1} \rightarrow q_{t+2} \rightarrow \cdots \rightarrow q_{t+p}$$
$$q_t = q_{t+p};\ q_{t+1} = q_{t+p+1}; \ldots; q_{t+p} = q_{t+2p}$$

We defined two constant configurations $\overline{0}$ and $\overline{1}$ as: $\overline{0} = [0, 0, \ldots, 0] \in \mathcal{S}^n$; $\overline{1} = [1, 1, \ldots, 1] \in \mathcal{S}^n$.

## 4 1nCA rules classification

In the following section a classification of the 1nCA Rules is presented. A central issue in the theory of cellular automata is the classification, i.e. understanding how cellular automata can be meaningfully grouped according to their structure and behavior. There are mainly two approach for the classification of the cellular automata: the direct way, called Phenotypic Classification, to classified cellular automata is to observe their behavior through the spatial-temporal patterns they generates out of several random initial conditions, and then to use statistical metrics to quantify the observed behavior (Li et al. 1990). Another approach, called Genotypic Classification, is based on the analysis of the automaton transition rules.

There are several works (e.g., Gutowitz et al. 1987; Li and Packard 1990; Sutner 1990; Wolfram 1983; Wuensche 1999) focusing on the classification of the one dimensional cellular automata and in particular on the Elementary Cellular Automata. In this section we present an approach of genotypic classification applied to the 1nCA. The idea of a genotypic classification of cellular automata is to divide a population of automata into groups according to the intrinsic properties of the rules. The aim is that some features of the cellular automata behaviors are predictable on the basis of a genotypic classification.

### 4.1 Totalistic rules

A cellular automaton is called *totalistic* if the value of a cell depends only on the sum of the values of its neighbors at the previous time step, and not on their individual values (Wolfram 1983). Therefore, half of the possible rules for 1nCA are *totalistic*. The sum $n$ of the neighborhood cells is computed $n = c_i + n_{c_i}^{(t)}$ and $0 \leq n \leq 2$. The following rules are totalistic:

| | |
|---|---|
| Rule 0Δ | $f(n) = 0$ |
| Rule 1Δ | $f(n) = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{if } n = 1 \\ 0 & \text{if } n = 2 \end{cases}$ |
| Rule 6Δ | $f(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 2 \end{cases}$ |
| Rule 7Δ | $f(n) = \begin{cases} 1 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 2 \end{cases}$ |
| Rule 8Δ | $f(n) = \begin{cases} 0 & \text{if } n = 0 \\ 0 & \text{if } n = 1 \\ 1 & \text{if } n = 2 \end{cases}$ |
| Rule 9Δ | $f(n) = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{if } n = 1 \\ 1 & \text{if } n = 2 \end{cases}$ |
| Rule 14Δ | $f(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ 1 & \text{if } n = 2 \end{cases}$ |
| Rule 15Δ | $f(n) = 1$ |

### 4.2 Neighbor-independent and self-independent

A rule is *Neighbor-Independent* if the value of a cell depends only on its own previous value and not on the value of the neighbors. Formally, a rule is *Neighbor-Independent* if $\forall s \in \mathcal{S}, f(s, 0) = f(s, 1)$ so, according to the definition of the transition function, a rule is *Neighbor-Independent* if $\beta_0 = \beta_1$, $\beta_2 = \beta_3$

A rule is *Self-Independent* if the value of a cell depends only on the value of the neighbors and not on its own previous value. Formally, a rule is *Self-Independent* if $\forall s \in \mathcal{S}, f(0, s) = f(1, s)$ so, according to the definition of the transition function, a rule is *Self-Independent* if $\beta_0 = \beta_2$, $\beta_1 = \beta_3$.

According to this criterion the 1nCA rules can therefore be characterized as follows:

Neighbor-Independent rules: Rule 0Δ, Rule 3Δ, Rule 12Δ, Rule 15Δ;
Self-independent rules: Rule 0Δ, Rule 5Δ, Rule 10Δ, Rule 15Δ.

**Table 1** The 1nCA rules characterized according to the $\lambda$ parameter

| | |
|---|---|
| $\lambda = 0$ | Rule 0Δ |
| $\lambda = 0.25$ | Rule 1Δ, Rule 2Δ, Rule 4Δ, Rule 8Δ |
| $\lambda = 0.5$ | Rule 3Δ, Rule 5Δ, Rule 6Δ, Rule 9Δ, Rule 10Δ, Rule 12Δ |
| $\lambda = 0.75$ | Rule 7Δ, Rule 11Δ, Rule 13Δ, Rule 14Δ |
| $\lambda = 1$ | Rule 15Δ |

### 4.3 $\lambda$-Parameter

An even cruder piece of information about a rule is the number of non-quiescent outputs in a rule-table. For the 1nCA this parameter is equal to the number of $\beta$ parameters that are equal to one and it can be calculated as $c = \sum_{i=0}^{3} \beta_i$

Langton (1990) proposed the so called $\lambda$-parameter as an order-chaos parameter for CA. This parameter measures the density of non-quiescent (not zero) outputs in a rule-table. For the 1nCA the $\lambda$-parameter can be calculated as: $\lambda = \frac{c}{k^n} = \frac{1}{4}\sum_{i=0}^{3} \beta_i$ where $k$ is the number of states and $n$ is the neighborhood size. $\lambda$ varies between 0 (order) to 0.5 (chaos) to 1 (order). As $\lambda$ is increased from 0 to 0.5 (or decreased from 1 to 0.5), the automata move from having the most homogeneous rule tables to having the most heterogeneous.

Langton presented evidence that there is some correlation between the $\lambda$ parameter and the behavior of an "average" Cellular Automata on an "average" initial configuration (Langton 1990): the behavior was characterized in terms of quantities such as single-site entropy, two-site mutual information, difference-pattern spreading rate, and average transient length. Generally the correlation is quite good for very low and very high $\lambda$ values, which predict fixed-point or short-period behavior. However, for intermediate $\lambda$ values, there is a large degree of variation in behavior (Mitchell et al. 1993).

The values of the $\lambda$ parameter for all the 1nCA rules are shown in Table 1.

**Table 2** The values of the *sensitivity parameter* for the 1nCA rules

| Rule | $\mu$ | Rule | $\mu$ | Rule | $\mu$ | Rule | $\mu$ |
|---|---|---|---|---|---|---|---|
| Rule 0Δ | 0 | Rule 1Δ | 0.5 | Rule 2Δ | 0.5 | Rule 3Δ | 0.5 |
| Rule 4Δ | 0.5 | Rule 5Δ | 0.5 | Rule 6Δ | 1 | Rule 7Δ | 0.5 |
| Rule 8Δ | 0.5 | Rule 9Δ | 1 | Rule 10Δ | 0.5 | Rule 11Δ | 0.5 |
| Rule 12Δ | 0.5 | Rule 13Δ | 0.5 | Rule 14Δ | 0.5 | Rule 15Δ | 0 |

**Table 3** The values of the *rule density* for all the rules

| Rule | $R\rho$ | Rule | $R\rho$ |
|---|---|---|---|
| Rule 0Δ | 0 | Rule 1Δ | 0.375 |
| Rule 2Δ | 0.25 | Rule 3Δ | 0.5 |
| Rule 4Δ | 0.25 | Rule 5Δ | 0.5 |
| Rule 6Δ | 0.5 | Rule 7Δ | 0.625 |
| Rule 8Δ | 0 | Rule 9Δ | 0.5 |
| Rule 10Δ | 0.5 | Rule 11Δ | 0.75 |
| Rule 12Δ | 0.5 | Rule 13Δ | 0.75 |
| Rule 14Δ | 1 | Rule 15Δ | 1 |

### 4.4 Sensitivity

Binder (1993, 1994) proposed the *sensitivity parameter* $\mu$, motivated by the observation that the Wolfram classes are characterized by its sensitivity to changes in the state of a unique cell of the neighborhood of the transition rule.

Sensitivity is defined as the number of changes in the outputs of the transition rule, caused by changing the state of each cell of the neighborhood, one cell at a time, over all possible neighborhoods of the rule being considered: $\mu = \frac{1}{nm}\sum_n \sum_{j=1}^{m} \frac{\delta f}{\delta s_j}$ where $m$ is the number of cells in the neighborhood and $n$ is the number of possible neighborhoods in the rule table. For 1nCA, $m = 2$, and $n = 2^m = 4$. The Boolean derivate for Cellular Automata (Vichniac 1990) $\frac{\delta f}{\delta s_j}$ is equal to 1 if $f(s_1, \ldots, s_j, \ldots) \neq f(s_1, \ldots, \neg s_j, \ldots)$, otherwise is equal to 0.

**Table 4** The rules divided according to the symmetries

The value of *rule density* is reported for each rule. The classes marked with **T** are formed by totalistic rules, **N** by Neighbor-Independent rules, and **S** by Self-Independent rules

| | | | | |
|---|---|---|---|---|
| Class 0ΔTNS: | Rule 0Δ | ($R\rho = 0$) | Rule 15Δ | ($R\rho = 1$) |
| Class 1ΔT: | Rule 1Δ | ($R\rho = 0.375$) | Rule 7Δ | ($R\rho = 0.625$) |
| Class 2Δ: | Rule 2Δ | ($R\rho = 0.25$) | Rule 11Δ | ($R\rho = 0.75$) |
| Class 3ΔN: | Rule 3Δ | ($R\rho = 0.5$) | | |
| Class 4Δ: | Rule 4Δ | ($R\rho = 0.25$) | Rule 13Δ | ($R\rho = 0.75$) |
| Class 5ΔS: | Rule 5Δ | ($R\rho = 0.5$) | | |
| Class 6ΔT: | Rule 6Δ | ($R\rho = 0.5$) | Rule 9Δ | ($R\rho = 0.5$) |
| Class 8ΔT: | Rule 8Δ | ($R\rho = 0$) | Rule 14Δ | ($R\rho = 1$) |
| Class 10ΔS: | Rule 10Δ | ($R\rho = 0.5$) | | |
| Class 12ΔN: | Rule 12Δ | ($R\rho = 0.5$) | | |

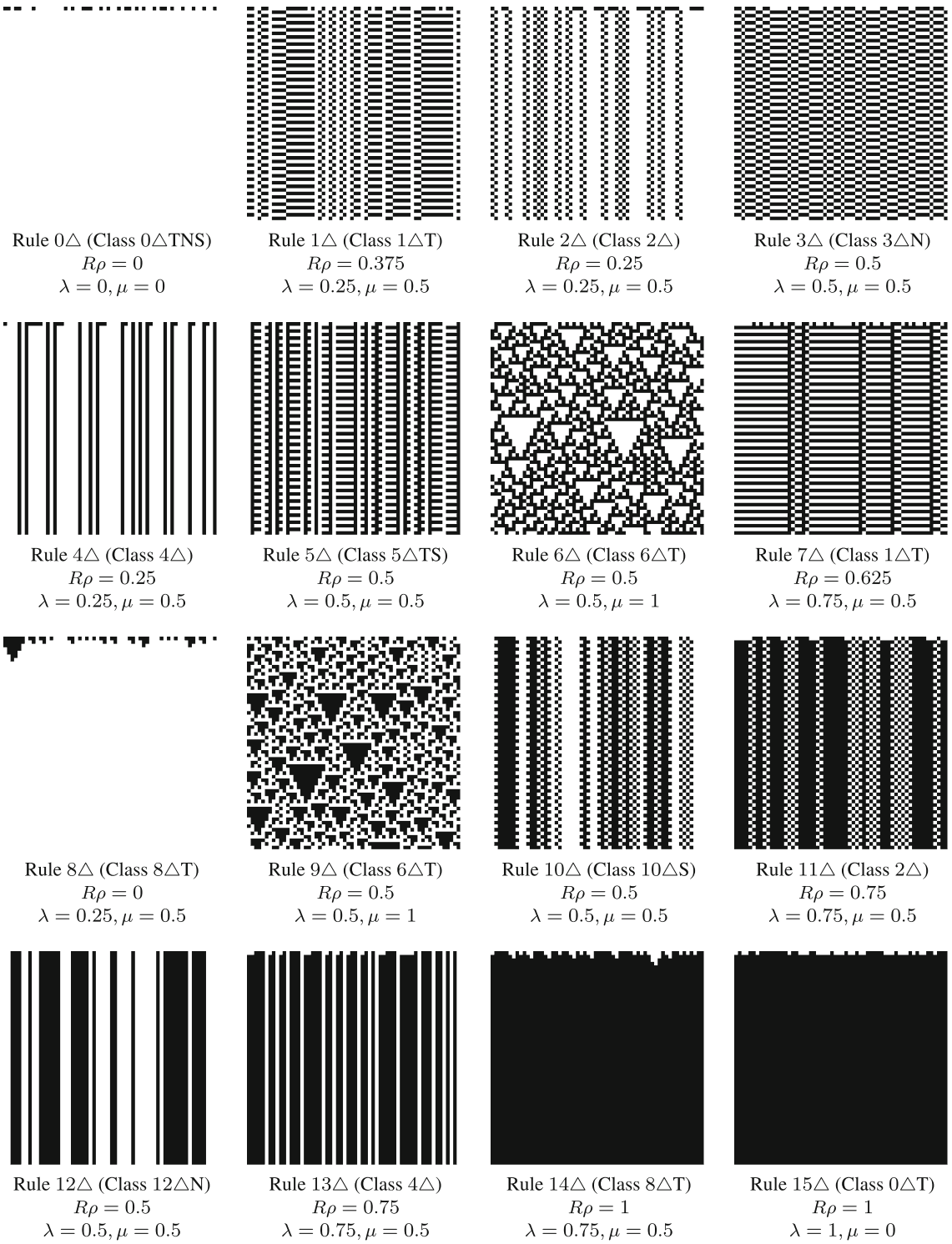


**Fig. 9** 60 steps of the time evolution of all the 16 1nCA with the default synchronous update scheme and periodic boundaries conditions starting from an initial random configuration of 60 cells

The sensitivity parameter takes on three different values: 0, 0.5, and 1. The sensitivity parameter helps to relatively discriminate null and chaotic behaviors: the null behavior happens in rules with low sensitivity and the chaotic behavior happens in rules with high sensitivity. Fixed-point and periodic behaviors are concentrated around 0.5.

The value of the sensitivity parameter for the 1nCA rules is presented in Table 2.

### 4.5 Rule density

The rule density is another very simple parameter introduced to describe rules behavior. The rule density, $R\rho$, is computed as $R\rho = (\lambda - \frac{1}{2})2^{(\beta_3 - \beta_0)} + \frac{1}{2}$.

Roughly speaking, rule density indicates the average fraction of sites whose value is one in the rule dynamic evolution. The rule density value is comprised between zero and one. A value of zero indicates that a rule converges (for most of the initial configurations) to zero state in all the cells, a value of one indicates a convergence to one. The rule density value for the 1nCA rules is shown in Table 3.

### 4.6 Rules symmetries

One mean of verification of the consistence of the rule density parameter (and also the other parameters) is the use of symmetries: if two rules are *conjugate*, the rule density of one rule is equals to $1 - R\rho$ of the other rule.

In Fatès (2003) the author defines the reflected, conjugate and reflected conjugate symmetries for the Elementary Cellular Automata. The only possible symmetry for the 1nCA is when $f^*$, the conjugate rule of $f$, is defined as

$$\forall (c_i, n_{c_i}) \in \mathcal{S}^2, f^*(c_i, n_{c_i}) = f(\neg c_l, \neg n_{c_i})$$

where $\neg$ denotes the operation of changing the zeros into ones and ones into zeros. The $\beta^*$ parameters of the conjugate rule are defined as $\beta_3^* = \neg\beta_0$; $\beta_2^* = \neg\beta_1$; $\beta_1^* = \neg\beta_2$; $\beta_0^* = \neg\beta_3$.

We identified 6 classes of rules, shown in Table 4, according to the symmetries: we can group in one class all the rules that are symmetric (reflected, conjugated or reflected conjugated). The classes are named according to the lowest member index. Each class is formed by totalistic or non-totalistic rules.

This kind of classification of the 1nCA is important because we can restrict the study of the dynamic behavior to only one member of each class and the behavior of the other members can be simply inferred according to the symmetric relations.

## 5 Effects of asynchronicity in 1nCA

In this section we present the effects of several relevant update schemes on 1nCA automata dynamic evolutions. In Fig. 9 the time evolutions of all the 16 rules according with synchronous update scheme and periodic boundaries conditions, starting from an initial random configuration of 60 cells.

We tested the following update schemes on all the 1nCA rules; more precisely, we analyzed the behavior of representatives of the symmetry classes identified in Sect. 4.6:

- Synchronous
- Random cyclic
- Equal frequency clocked
- Random order
- Random independent

In the following section the outcomes of this analysis will be shown for rules belonging to rules of class 6ΔT, given their particular sensitiveness to the choice of a different update scheme and to the significant difference in the overall system dynamics. A summary of the results of the overall analysis will be given in Sect. 5.2.

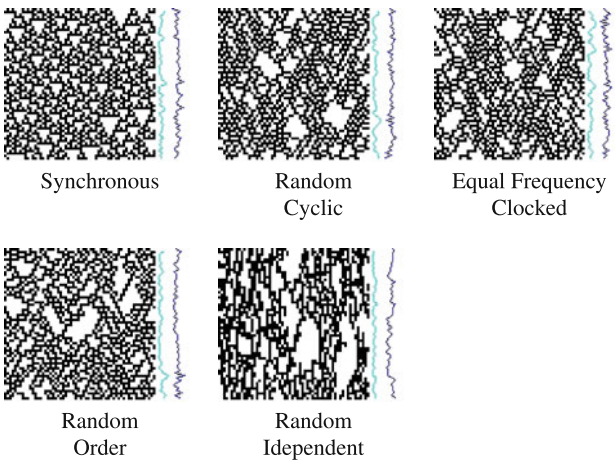


**Fig. 10** Time space diagrams of Rule 6Δ using different update schemes starting from a random initial configuration

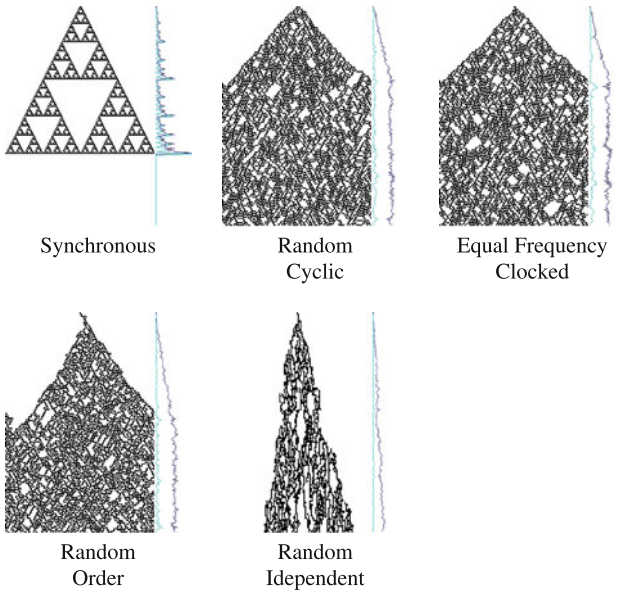


**Fig. 11** Time space diagrams of Rule 6Δ using different update schemes starting from a single seed

**Table 5** Summary of 1nCA rules classification according both to the

| Rule | Bin | Wolfram class | Li-Packard class | Class | $\lambda$ | $R\rho$ | $\mu$ | Asynchrony sensitive |
|---|---|---|---|---|---|---|---|---|
| 0Δ | 0000 | W1 | Null | 0ΔTNS | 0 | 0 | 0 | AS1 |
| 1Δ | 0001 | W2 | Two-cycle | 1ΔT | 0.25 | 0.375 | 0.5 | AS3 |
| 2Δ | 0010 | W2 | Two-cycle | 2Δ | 0.25 | 0.25 | 0.5 | AS4 |
| 3Δ | 0011 | W2 | Two-cycle | 3ΔN | 0.5 | 0.5 | 0.5 | AS2 |
| 4Δ | 0100 | W1 | Fixed-point | 4Δ | 0.25 | 0.25 | 0.5 | AS1 |
| 5Δ | 0101 | W2 | Two-cycle | 5ΔS | 0.5 | 0.5 | 0.5 | AS3 |
| 6Δ | 0110 | W3 | Chaotic | 6ΔT | 0.5 | 0.5 | 1 | AS4 |
| 7Δ | 0111 | W2 | Two-cycle | 1ΔT | 0.75 | 0.625 | 0.5 | AS3 |
| 8Δ | 1000 | W1 | Null | 8ΔT | 0.25 | 0 | 0.5 | AS1 |
| 9Δ | 1001 | W3 | Chaotic | 6ΔT | 0.5 | 0.5 | 1 | AS4 |
| 10Δ | 1010 | W2 | Two-cycle | 10ΔS | 0.5 | 0.5 | 0.5 | AS3 |
| 11Δ | 1011 | W2 | Two-cycle | 2Δ | 0.75 | 0.75 | 0.5 | AS4 |
| 12Δ | 1100 | W2 | Fixed-point | 12ΔN | 0.5 | 0.5 | 0.5 | AS1 |
| 13Δ | 1101 | W2 | Fixed-point | 4Δ | 0.75 | 0.75 | 0.5 | AS1 |
| 14Δ | 1110 | W1 | Null | 8ΔT | 0.75 | 1 | 0.5 | AS1 |
| 15Δ | 1111 | W1 | Null | 0ΔTNS | 1 | 1 | 0 | AS1 |

### 5.1 Class 6ΔT

This section describes an analysis of the dynamic evolution of 1nCA belonging to the Class 6Δ. The rules of this class are Rule 6Δ ($R\rho = 0.5$, $\lambda = 0.5$, $\mu = 1$) and Rule 9Δ ($R\rho = 0.5$, $\lambda = 0.5$, $\mu = 1$).

The dynamic behaviors of the Rule 6Δ using different update schemes are shown in Fig. 10. The rules of this class are *Chaotic*: these rules are characterized by the exponential divergence of its cycle length with the system size, and for the instability with respect to perturbations. If the number of cells is finite, for the Synchronous, Random Cyclic, and Equal Frequency Clocked schemes, the evolution eventually falls into a cycle (with period $p > 1$) or a fixed point ($p = 1$). The configuration $\overline{0}$ is the fixed point of the Rule 6Δ, the configuration $\overline{1}$ is the fixed point of the Rule 9Δ. These configurations are fixed points also using the Random update schemes.

Changing update scheme has dramatic effect on the rule of this class. As shown in Fig. 11, with the Synchronous update scheme, the Rule 6Δ produces a dynamic evolution similar to the *Sierpinski Triangle* fractal. This typical shape is not present with any of the other update schemes.

Moreover if the automaton has periodic boundaries conditions and the number of cells is a power of two, starting from an initial configuration, the evolution of the synchronous automata eventually reaches the fixed point. The automata with the other update schemes does not present this behavior.

### 5.2 Summary of effects of asynchronicity in 1nCA

We can classify the 1nCA rules according to the impact of the choice of a different update schemes on the dynamic evolution of the model. Comparing the dynamic behavior of the same automata adopting the synchronous update scheme and a different asynchronous update scheme, we identified the following four classes of asynchrony influence:

- *AS1*: not influenced by the update schemes, the dynamic behavior does not change varying the update scheme;
- *AS2*: the Random Independent update scheme perturbs the dynamic behavior;
- *AS3*: the Random Independent and Random Order update schemes perturb the dynamic behavior;
- *AS4*: any asynchronous update scheme perturbs the dynamic behavior.

The 1nCA rules can be classified into the above categories as shown in Table 5: the table also reports the rule classification according to existing approaches [and more precisely Wolfram's and Li and Packard's (1990) classifications], as well as to other classifications introduced in Sect. 4.

## 6 Conclusions

The paper has presented a discussion on asynchronicity in CA models, comparing different types of update scheme

and proposing an ontology to classify them. The implications of the different update schemes have been presented by introducing a very simple CA based model and testing it adopting different update schemes. Future developments of this work, in the vein of Fatès and Morvan (2005), are aimed at evaluating the possibility to define asynchronous models in which some global dynamic properties are preserved even adopting different asynchronous update schemes.

**Acknowledgment** The work presented in this paper has been partially funded by the University of Milano-Bicocca within the project "Fondo d'Ateneo per la Ricerca - anno 2010".

## References

Bandini S, Manzoni S, Vizzari G (2005) Situated agents interaction: coordinated change of state for adjacent agents. In: Malyshkin VE (ed) PaCT, Lecture notes in computer science, vol 3606. Springer, Berlin, pp 114–128

Bandini S, Vizzari G (2006) Regulation function of the environment in agent-based simulation. In: Weyns D, Parunak HVD, Michel F (eds) E4MAS, Lecture notes in computer science, vol 4389. Springer, Berlin, pp 157–169

Binder P (1993) A phase diagram for elementary cellular automata. Complex Syst 7:241–247

Binder P (1994) Parametric ordering of complex systems. Phys Rev E 49(3):2023–2025

Cornforth D, Green DG, Newth D (2005) Ordered asynchronous processes in multi-agent systems. Physica D 204(1–2):70–82

Darabos C, Giacobini M, Tomassini M (2007) Semi-synchronous activation in scale-free boolean networks. In: e Costa FA, Rocha LM, Costa E, Harvey I, Coutinho A (eds) ECAL, Lecture notes in computer science, vol 4648. Springer, Berlin, pp 976–985

Fang L, Antsaklis P, Tzimas A (2005) Asynchronous consensus protocols: preliminary results, simulations and open questions. In: Decision and control, 2005 and 2005 European control conference. CDC-ECC '05. 44th IEEE Conference, pp 2194–2199

Fatès N (2003) Experimental study of elementary cellular automata dynamics using the density parameter. In: Morvan M, Rémila (eds) Discrete models for complex systems, DMCS'03, DMTCS proceedings. Discrete mathematics and theoretical computer science, vol AB, pp 155–166

Fatès N, Morvan M (2005) An experimental study of robustness to asynchronism for elementary cellular automata. Complex Syst 16(1):1–27

Gutowitz H, Victor JD, Knight BW (1987) Local structure theory for cellular automata. Physica D 28:18–48

Kanada Y (1994) The effects of randomness in asynchronous 1d cellular automata (poster). Artificial Life IV

Langton CG (1990) Computation at the edge of chaos. Physica D 42:12–37

Li W, Packard N (1990) The structure of the elementary cellular automata rule space. Complex Syst 4(3):281–297

Li W, Packard N, Langton CG (1990) Transition phenomena in CA rule space. Physica D 45:77

Mitchell M, Hraber PT, Crutchfield JP (1993) Revisiting the edge of chaos: evolving cellular automata to perform computations. Complex Syst 7:89–130

Page SE (1997) On incentives and updating in agent based models. Comput Econ 10:67–87

Paolo EAD (2000) Searching for rhythms in asynchronous random boolean networks. In: Bedau M (ed) Alife VII: proceedings of the seventh international conference, MIT Press, Cambridge, pp 73–80

Schönfisch B, de Roos A (1999) Synchronous and asynchronous updating in cellular automata. Biosystems 51(3):123–143

Sutner K (1990) Classifying circular CA. Physica D 45:386

Vichniac GY (1990) Boolean derivatives on cellular automata. Phys D 45(1–3):63–74

Wolfram S (1983) Cellular automata. Los Alamos Sci 9:2–21

Wolfram S (1983) Statistical mechanics of cellular automata. Rev Mod Phys 55:601–644

Wuensche A (1999) Classifying cellular automata automatically: finding gliders, filtering, and relating space-time patterns, attractor basins, and the Z parameter. Complexity 4(3):47–66