Chaos, Solitons and Fractals 113 (2018) 209–220

Contents lists available at ScienceDirect

Chaos, Solitons and Fractals

Nonlinear Science, and Nonequilibrium and Complex Phenomena

journal homepage: www.elsevier.com/locate/chaos

Frontiers

Characterisation of the elementary cellular automata in terms of their maximum sensitivity to all possible asynchronous updates

Eurico L.P. Ruivo[a,*], Marco Montalva-Medel[b], Pedro P.B. de Oliveira[a], Kévin Perrot[c]

[a] Universidade Presbiteriana Mackenzie, Faculdade de Computação e Informática, São Paulo, SP, Brazil
[b] Universidad Adolfo Ibáñez, Facultad de Ingeniería y Ciencias, Santiago, Chile
[c] Aix Marseille Univ., Univ. de Toulon, CNRS, LIS, UMR 7020, Marseille, France

ARTICLE INFO

Article history:
Received 20 December 2017
Revised 5 June 2018
Accepted 8 June 2018
Available online 23 June 2018

Keywords:
Cellular automaton
Asynchronous update
Update digraph
Discrete dynamics
One-step maximum sensitivity

ABSTRACT

Cellular automata are fully-discrete dynamical systems with global behaviour depending upon their locally specified state transitions. They have been extensively studied as models of complex systems as well as objects of mathematical and computational interest. Classically, the local rule of a cellular automaton is iterated synchronously over the entire configuration. However, the question of how asynchronous updates change the behaviour of a cellular automaton has become a major issue in recent years. Here, we analyse the elementary cellular automata rule space in terms of how many different one-step trajectories a rule would entail when taking into account all possible deterministic ways of updating the rule, for one time step, over all possible initial configurations. More precisely, we provide a characterisation of the elementary cellular automata, by means of their one-step maximum sensitivity to all possible update schedules, that is, the property that any change in the update schedule causes the rule's one-step trajectories also to change after one iteration. Although the one-step maximum sensitivity does not imply that the remainder of the time-evolutions will be distinct, it is a necessary condition for that.

1. Introduction

Cellular automata (CAs) are locally-defined dynamical systems, discrete with respect to time, space and state variables. They have been studied both from the point of view of their mathematical and computational properties as well as systems capable of simulating real-world phenomena [1], such as disease spread [2], urban growth [3] and fluid dynamics [4]. Even with simply defined local interaction rules, cellular automata may be capable of displaying arbitrarily complex global emergent behaviour; well-known binary cellular automata rules such as *The Game of Life* [5] and elementary cellular automaton *Rule 110*, are known to be capable of simulating a Turing machine [6].

Classically, the time-evolution of a configuration under a cellular automaton local rule is given by synchronously iterating such a rule over the entire configuration. However, the study of the effects of allowing asynchronous updates has been gaining attention in recent years. By taking into account additional update schemes further to the synchronous, the rules typically unveil a richer behavioural set, both in terms of their possible dynamics and capabilities of problem-solving as well as of simulating real-world phenomena; see [7] for a comprehensive review on asynchronism and how it compares to the synchronous case.

From the viewpoint of simulating real-world complex systems, it is usually considered that they are more naturally simulated by asynchronous cellular automata rather than synchronous ones [8], since they generally result from the parts of the system interacting asynchronously based upon action and reaction.

On the other hand, from the computational point of view, some dynamical properties of certain cellular automata rules are partially due to the synchronism rather than due to only the rule itself [9,10], what leads to the natural question of which might be the new capabilities of a rule when allowing asynchronous update schedules [11].

In this context, a naturally relevant question is to understand how changes in the way a rule is updated over the set of configurations of a given length affects, both quantitatively and qualitatively, its dynamical behaviour. Aracena et al. [12] have constructively shown that it is possible to partition the set of possible updates into equivalence classes. The quantity of such classes may be significantly smaller than the number of individual updates in many situations, thus rendering comprehensive studies of all possible updates more tractable in these situations.

One particular way to study asynchronous binary cellular automata is to use the more general framework of *Boolean automata*

* Corresponding author.
E-mail address: eurico.ruivo@mackenzie.br (E.L.P. Ruivo).

https://doi.org/10.1016/j.chaos.2018.06.004
0960-0779/

networks (BANs), where the asynchronism is coded in the structure of labeled digraphs named *interaction digraphs*. Many dynamical properties, such as the quantification of fixed points [13,14] and the existence of limit-cycles [15], have been studied in such a context.

Here, we address the question of how changing the update schedule affects the number of different dynamics of a cellular automaton, after one iteration of the rule. More precisely, the term *dynamics* here refer to the set of pairs of configurations and their respective image under the CA rule under a given update schedule. In order to do so, we provide a way to compute how many independent update schedules do exist for configurations of a given length, establish the notion of one-step update schedule sensitivity and give necessary and sufficient conditions for an elementary cellular automaton to have maximum one-step sensitivity to changes in updates. We constrain to the well studied family of elementary cellular automata, where each cell can take on two possible states and the state transition of every cell of an automaton depends upon the states of the cell itself and of its immediate neighbours to the left and to the right. We understand that our short-range, one-iteration based study is a key step towards better understanding the longer-range dynamical behaviour of one-dimensional cellular automata, possibly even their limit behaviours.

In order to know how much the dynamics of a rule varies for different update schedules, the notion of *update schedule sensitivity* is defined. We believe that such a measure may open new possibilities to study classical problems in the CA literature, previously addressed only in the synchronous update schedule; for instance, the *density classification task*, for which its has been shown that no single CA rule is able to solve it under synchronous update schedule (see [16] for a survey on this problem). The question of whether or not there might be a solution for some asynchronous update schedule is still an open question. Naturally, CAs with many different dynamics due to distinct update schedules, offers many more possibilities to tackle the problem.

Basically, two update schedules are said to be equivalent if for any CA rule and any configuration, the image of the configuration generated by iterating the CA rule under the first update schedule is the same as the one obtained by iterating the CA rule under the second update schedule. Also, a CA rule is said to have *maximum (one-step) update schedule sensitivity* when, for any two non-equivalent update schedules, there will be a configuration for which its image will be distinct for each update schedule. This will be addressed more formally in Section 4, but it is worth showing how this one-step property may affect the global behaviour of the rule along time.

First of all, it is worth noticing that even if the images of a configuration under distinct update schedules may differ in one iteration of the rule, it is possible that after a number iterations the trajectories become the same.

However, the one-step difference is a necessary condition in order for the trajectories of a configuration under distinct update schedules to be distinct, hence checking a rule for maximum one-step update schedule sensitivity is, in reality, telling whether or not a rule is capable of displaying distinct time-evolutions for small changes in a given update schedule.

In order to illustrate the above possibilities, Fig. 1 shows the difference of time-evolutions of ECA rules 110 (well-known for being Turing-universal for synchronous update) and 232 (the majority rule) for two almost equal update schedules: the synchronous one (left) and the one in which the leftmost cell is updated before all othere cells (middle), followed by the difference of the time-evolutions (right). Therefore, as said above, maximum update schedule sensitivity is not sufficient, but it is necessary, in order for a rule to present long-term differences in the time-evolution in response for small changes in the update schedule.

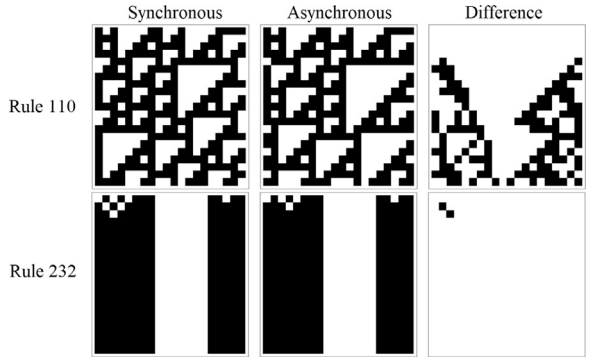

Fig. 1. Time-evolution ECA rules 110 and 232 for the synchronous update schedule (left) and an asynchronous update schedule where the leftmost cell is updated before the other cells (middle), where time flows downwards and white and black cells represent cells in state 0 and 1, respectively. A one-step difference in the time-evolutions may (rule 110) or may not (rule 232) result in distinct trajectories after a given number of time-steps: the diagrams on the right represent the differences between each pair of time-evolutions (right).

The paper is organised as follows: the next section provides basic definitions regarding cellular automata, general update schedules and update digraphs. Section 3 summarises preliminary results regarding the maximum number of distinct updates schedules and of the distinct dynamics for configurations of a given length. As follows, Section 4 defines the notion one-step sensitivity to the update schedules, and Section 5 then gives a characterisation in terms of the local rules of the elementary cellular automata that display maximum one-step sensitivity to all possible update schedules. Finally, concluding remarks are made in Section 6.

2. Basic definitions

2.1. Cellular automata, configurations and dynamical equivalence

A *cellular automaton* (CA) is a quadruple (S, N, f, d), where $S = \{0, 1, \cdots, k-1\}$ is the *state set*, $N = (\vec{n}_1, \vec{n}_2, \cdots, \vec{n}_m) \in (\mathbb{Z}^d)^m$ is the *neighbourhood vector*, $f : S^m \longrightarrow S$ is the *local function* (or *local rule*) and $d \in \mathbb{Z}_+$ is the *dimension*. In particular, one-dimensional binary CAs have $S = \{0, 1\}$, $d = 1$ and, given $r \in \{\frac{m}{2} : m \in \mathbb{Z}_+\}$, the rule is said to be *radius-r* when $N = (-\lceil r \rceil, -\lceil r \rceil + 1, \cdots, 0, \cdots, \lfloor r \rfloor - 1, \lfloor r \rfloor)$.

An *elementary cellular automaton* (ECA) is a radius-1, binary, one-dimensional cellular automaton. There are 256 ECAs arising from the distinct possible local rules $f : \{0, 1\}^3 \longrightarrow \{0, 1\}$; we refer to the set of all ECA rules by $\mathscr{F}$. Each ECA local rule f may be identified by its *Wolfram number* [17] given by

$$W(f) = \sum_{(q_1,q_2,q_3)\in\{0,1\}^3} f(q_1, q_2, q_3)2^{(2^2q_1+2^1q_2+2^0q_3)}$$

A (periodic) *configuration of length L* is a function $c : \mathbb{Z}_L \longrightarrow \{0, 1\}$ where $\mathbb{Z}_L$ denotes the set of integers modulo L. Each index $i \in \mathbb{Z}_L$ of a configuration is named a *cell* and $c(i)$, denoted by c_i from now on, is the *state* of cell i. For the ECA rules, the vector $(i-1, i, i+1)$ is the *neighbourhood* of cell i and cells $i-1$, i and $i+1$ are the *neighbours* of cell i. Here we denote the vector (c_{i-1}, c_i, c_{i+1}) by $[c_i]$. We denote the set of all periodic configurations of length L by $\mathscr{C}_L$ and a periodic configuration of length L simply by the vector $(c_1, c_2, \cdots, c_L)$ characterising $c(1)$ to $c(L)$.

Given a local ECA rule f, it induces a *synchronous global function* (or synchronous global rule) $F : \mathscr{C}_L \longrightarrow \mathscr{C}_L$, such that $(F(c))_i = f(c_{i-1}, c_i, c_{i+1}) = f([c_i])$. That is, the synchronous global rule results from applying the local rule f to the neighbourhood of each cell i, synchronously.

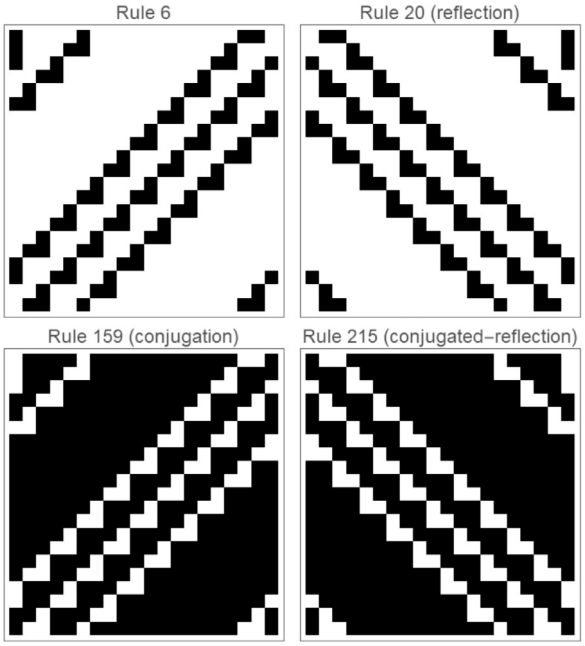

Fig. 2. Time-evolution of dynamically equivalent ECA rules 6, 20, 159 and 215, over symmetrical configurations. The dynamical behaviour of the rules are qualitatively the same. White and black cells represent states 0 and 1 respectively and time flows downwards.

Given a configuration c, the *conjugate of c*, the *reflection of c* and the *reflected-conjugation* (or *conjugated-reflection*) of c are, respectively, the configurations $\overline{c}$, c' and $\overline{c}'$ given by:

1. $\overline{c}_i = 1 - c_i$
2. $c'_i = c_{-i}$
3. $\overline{c}'_i = 1 - c_{-i}$

In other words, $\overline{c}, c'$ and $\overline{c}'$ are obtained from c by swapping 0s and 1s, by reversing left and right and by reversing left and right and swapping 0s and 1s at the same time, respectively.

Instead of studying the whole ECA rule space, one may consider only a representative of each *dynamical equivalence class*. Two rules are dynamically equivalent if they have the same qualitative dynamical behaviour except possibly for conjugation (swapping 0s for 1s in a configuration), reflection (reading a configuration from right to left instead of from left to right) or conjugated-reflection (applying a reflection and a conjugation to a configuration).

More formally, two local ECA rules f and g are *dynamically equivalent* if one of the following occurs:

1. $f(q_1, q_2, q_3) = 1 - g(1 - q_1, 1 - q_2, 1 - q_3)$, for all $(q_1, q_2, q_3) \in \{0, 1\}^3$ (conjugation);
2. $f(q_1, q_2, q_3) = g(q_3, q_2, q_1)$, for all $(q_1, q_2, q_3) \in \{0, 1\}^3$ (reflection);
3. $f(q_1, q_2, q_3) = 1 - g(1 - q_3, 1 - q_2, 1 - q_1)$, for all $(q_1, q_2, q_3) \in \{0, 1\}^3$ (reflected-conjugation).

In fact, each of the above conditions on the local rules reflect in the global rules respectively as follows:

1. $\overline{F(c)} = G(\overline{c})$;
2. $\left(F(c)\right)' = G(c')$;
3. $\left(\overline{F(c)}\right)' = G(\overline{c}')$.

The ECA rule space is then partitioned in 88 classes of dynamical equivalent rules. Fig. 2 shows the evolution of ECA rules 6, 20, 159 and 215 (dynamically equivalent to each other) for the corresponding symmetrical configurations of the original, due to conjugation, reflection and conjugated-reflection.

2.2. Update schedules and update digraphs

In a more general approach, one may consider other ways to update each cell. For instance, given a finite configuration $c = (c_1, \cdots, c_L)$ it is possible to update it *sequentially*, that is, one cell after its preceding neighbour. In that case, one would compute $y_1 = f(c_L, c_1, c_2)$, updating cell 1, then $y_2 = f(y_1, c_2, c_3)$ to update cell 2, $y_3 = f(y_2, c_3, c_4)$ to update cell 3, and so on, until $y_L = f(y_{L-1}, c_L, y_1)$ to update cell L.

A particular (deterministic) way to update a periodic configuration can be referred to as an *update schedule*. Classically, cellular automata are studied with the synchronous update schedule, but there is a growing interest in how changing the update schedule affects the CA dynamics. An *update schedule* over the set of periodic configurations of lenght L is a function $s : \{1, \cdots, L\} \longrightarrow \{1, \cdots, L\}$ that tells the *priority* of each cell to be updated.

For instance, the update schedule $s : \{1, \cdots, L\} \longrightarrow \{1, \cdots, L\}$ given by $s(i) = \begin{cases} 1, & \text{if } i \text{ is odd} \\ 2, & \text{otherwise} \end{cases}$ indicates that the odd cells will be updated first (at the same time), and the even cells will be updated afterwards; also, for the synchronous update, it is given by $s(i) = 1$, for all $i \in \{1, \cdots, L\}$.

Here we focus on studying how sensitive ECA rules are to changes of the update schedules for finite, periodic configurations. In order to develop the ideas discussed forward in the text, it is convenient to establish a graph notation for the configurations and the interaction among cells. Bearing that in mind, it is necessary to give basic definitions on graphs and how it relates to the representation of update schedules for ECAs iterated over periodic configurations.

A *directed graph* (or *digraph*) is a pair $G = (V, E)$ where $V \subset \mathbb{Z}$ is a finite set of *vertices* and $E \subset V \times V$ is the set of *edges*. The first coordinate of an edge is an *initial vertex* and the last coordinate is a *terminal vertex*. A *path* in the digraph is a sequence $v_1, v_2, \cdots, v_n$ of vertices such that $(v_i, v_{i+1}) \in E$. In that case, the path is said to have *length* $n - 1$. A path $v_1, v_2, \cdots, v_n$ such that $v_i \neq v_j$ for $i \neq j$, except for $i = 1$ and $j = n$ is a *cycle* of length $n - 1$.

Here, we use a digraph with labeled edges to encode update schedules, as in [12,18]. The *ECA digraph of length L* is the digraph $G_L = (V, E)$ such that $V = \{1, \cdots, L\}$ and $(i, j) \in E$ if, and only if, $i \neq j$ and i is a neighbour of j. In other words, the vertices represent the indices of the cells in a configuration and there will be an edge from vertex i to vertex j if, and only if, i is the index of a cell in the neighbourhood of j but it is not the cell in position j itself. The idea encoded in the edge (i, j) is that a *change* in the state of cell i before cell j is updated will affect the neighbourhood of cell j and possibly the state transition of cell j itself.

That being said, it is now possible to encode an update schedule by labeling the edges of an ECA digraph. A *labeling* of a digraph $G = (V, E)$ is a function $lab : E \longrightarrow \{+, -\}$ associating a symbol + or - to each edge of G. Given an update schedule $s : \{1, \cdots, L\} \longrightarrow \{1, \cdots, L\}$ and an ECA digraph $G_L = (V, E)$ the *labeling induced by s* is the function $lab_s : E \longrightarrow \{+, -\}$ given by

$$lab_s(i, j) = \begin{cases} +, & \text{if } s(i) \geq s(j) \\ -, & \text{otherwise} \end{cases} .$$

In other words, if $e = (i, j) \in E$ and cell i is updated before cell j in s, then the edge e is labeled with a - ; if cell i is updated after time cell j or at the same time, then e is labeled with a + . The pair (G_L, lab_s) is an *ECA update digraph* of length L.

Although each update schedule s induces a unique labeling lab_s in a given digraph G, the converse is not true, i.e., given a labeling lab_{s_1} of G, there may exist an update schedule $s_2 \neq s_1$ such that $lab_{s_1} = lab_{s_2}$. Such an issue will be addressed later, after detailing

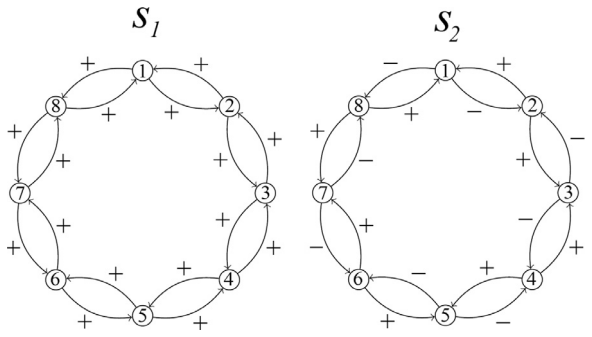

Fig. 3. Examples of labelings of an ECA update digraph of length 8, with s_1 being the synchronous update and s_2 the update where the odd cells are updated before the even cells.

how a local transition function and an update schedule induce a global transition function.

Fig. 3 shows two examples of labelings of an ECA update digraph of length 8 induced by the following update schedules:

$$s_1(i) = 1, \text{ for all } i \in \{1, \cdots, 8\} \quad s_2(i) = \begin{cases} 1, & \text{if } i \text{ is odd;} \\ 2, & \text{if } i \text{ is even.} \end{cases}$$

Let $s : V = \{1, \cdots, L\} \longrightarrow \{1, \cdots, L\}$ be an update schedule such that $s(V) = \{1, \cdots, l\}$, for some $l \leq L$. The *block with priority i*, with $1 \leq i \leq l$, is the set $B_i = \{j \in V : s(j) = i\}$. Let f be an ECA's local rule. Then $F_{B_i} : \mathscr{C}_L \longrightarrow \mathscr{C}_L$ is defined as:

$$\left(F_{B_i}(c)\right)_j = \begin{cases} f([c_j]), & \text{if } j \in B_i \\ c_j, & \text{otherwise} \end{cases}.$$

That is, in a given periodic configuration, F_{B_i} updates only the states of the cells j such that $s(j) = i$. The *ECA global function* $F_s : \mathscr{C}_L \longrightarrow \mathscr{C}_L$ *induced by f and s* is then given by

$$F_s = F_{B_l} \circ F_{B_{l-1}} \circ \cdots \circ F_{B_2} \circ F_{B_1}.$$

The *dynamics* of an ECA rule f with update schedule s over G_L is the set $D(G_L, f, s) = \{(c, F_s(c)) : c \in \mathscr{C}_L\}$. Given an ECA local rule f and an ECA digraph G_L, the set of dynamics of an ECA rule f is the set $D(G_L, f) = \{D(G_L, f, s) : s \text{ is an update schedule over } G_L\}$.

In the next section we present preliminary results on the relation between labelings, update schedules and the number of different dynamics of a rule.

3. Preliminary results and motivations

We say that two update schedules s_1 and s_2 over $\mathscr{C}_L$ are *dynamically equivalent* when $lab_{s_1} = lab_{s_2}$. In this case, Aracena et al. [19] have shown that both s_1 and s_2 have exactly the same dynamic, that is, $F_{s_1} = F_{s_2}$ for any global rule F. So that, given a labeling lab_s, one may consider any of the update schedules inducing lab_s without loss of generality.

As a consequence of this result, Aracena et al. [18] defined the *equivalence class* of an update schedule s over general update digraphs. Here, we discuss the particular case for update schedules over ECA update digraphs. The equivalence class of an update schedule s over $\mathscr{C}_L$ is the set

$$[s]_L = \{s : (G_L, lab_s) = (G_L, lab_s)\}.$$

This means that update schedules in the same equivalence class are dynamically equivalent, since those inducing the same labeling are dynamically equivalent, as shown in [19].

Let $U(G_L)$ be the set of all equivalence classes of update schedules over $\mathscr{C}_L$. $|U(G_L)|$ is an upper bound for the number of different dynamics that might arise from iterating a particular CA rule with all possible update schedules over $\mathscr{C}_L$ and such a bound depends only on the topology of G_L; in contrast, notice that this is not the case for $D(G, f)$, which also depends on the local function f.

In this context, Aracena et al. [12] worked out particular values of $|U(G)|$ for families of some digraphs G, such as complete digraphs and digraphs containing a tournament: digraphs in which there is a single directed edge connecting each pair of vertices. They are based on a characterisation for more general update digraphs that we also state here for the particular case of ECA update digraphs.

Theorem 1. *[18] A pair (G, lab) is an ECA update digraph if, and only if, $G = G_L$ and (G_R, lab_R) does not contain any forbidden cycle, where:*

- *(G_R, lab_R) is the* labeled reoriented multidigraph *(LRM) associated to the labeled digraph (G, lab), obtained by inverting the orientation of all the negative edges of (G, lab), but keeping its labels.*
- *A* forbidden cycle *in a labeled digraph is a cycle containing a negative edge.*

Notice that the above result refers to the *structure* of an ECA update digraph of length L but does not depend on any choice of an ECA rule in particular.

Fig. 4 shows examples of two labelings and their corresponding LRMs. Notice that, while (G^1, lab^1) is an ECA update digraph, (G^2, lab^2) is not, since its corresponding LRM, (G^2_R, lab^2_R), has one forbidden cycle: the one at the very top of the graph, made up of two negative edges.

In this way, given a digraph G_L, computing $|U(G_L)|$ becomes the combinatorial problem of counting the labelings of G_L not containing forbidden cycles in their LRMs.

In the remainder of the paper, we address the notion of *maximum* update schedule sensitivity.

4. Update Schedule Sensitivity

Let G_L be an ECA digraph of length L and f an ECA local rule. The *update schedule sensitivity* (UDSS) of the pair (G_L, f) is given by

$$\text{UDSS}(G_L, f) = \frac{|D(G_L, f)|}{|U(G_L)|},$$

where $|D(G_L, f)|$ is the number of possible dynamics for all possible update schedules in configurations of length L; so, this means that $\text{UDSS}(G_L, f)$ is the ratio of all the dynamics entailed by rule f, out of all possible dynamics in configurations of length L. Notice that update schedule sensitivity actually refers to a single step of application of the local function f, so that the concept should be more precisely characterised as a *one-step* sensitivity. For the sake of simplicity, we omit the 'one-step' attribute of the concept throughout the text; but we advise the reader to bear this in mind.

For instance, ECA rule 0, given by $f_0(q_1, q_2, q_3) = 0$, for all $(q_1, q_2, q_3) \in \{0, 1\}^3$ and any update schedule s over $\mathscr{C}_L$ yields the dynamics given by $D(G_L, f_0, s) = \{\big(c, (\underbrace{0, \cdots, 0}_{L \text{ times}})\big) : c \in \mathscr{C}_L\}$.

Hence,

$$\begin{aligned} |D(G_L, f_0)| &= |\{D(G_L, f_0, s) : s \text{ is an update schedule over } G_L\} \\ &= |\{\{\big(c, (\underbrace{0, \cdots, 0}_{L \text{ times}})\big) : c \in \mathscr{C}_L\}\}| = 1 \end{aligned}$$

and $\text{UDSS}(G_L, f_0) = 1/|U(G_L)|$, which is the minimum update schedule sensitivity.

On the opposite hand, some ECA rules do reach the maximum value of UDSS. In this case, we say a rule f has *maximum update schedule sensitivity* when $\text{UDSS}(G_L, f) = 1$, for all configurations of length $L \geq 3$. Putting it another way, rule f has maximum UDSS for

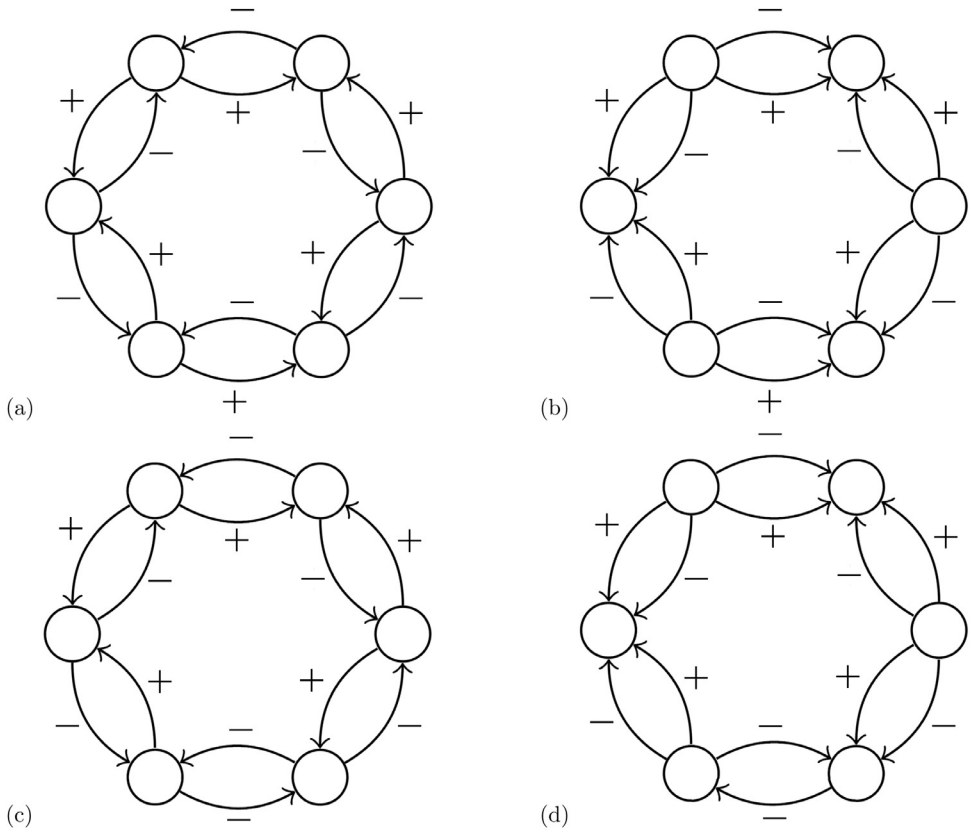

Fig. 4. (a) and (b): Labeled digraph (G^1, lab^1), which is an ECA update digraph, and its associated LRM, (G^1_R, lab^1_R). (c) and (d): Labeled digraph (G^2, lab^2), which is not an ECA update digraph, and its associated LRM, (G^2_R, lab^2_R).

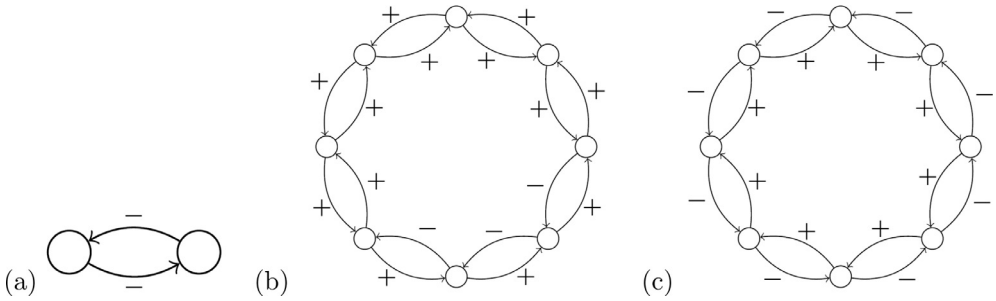

Fig. 5. (a) The only possible forbidden cycle of length 2 in $((G_L)_R, lab_R)$. (b) and (c) Examples with $L = 8$. A forbidden cycle of length 8 in $((G_L)_R, lab_R)$ with: (b) 3 negative labels in clockwise orientation; (c) 8 negative labels in counterclockwise orientation.

configurations of length L if, and only if, given any pair of non-equivalent update schedules s_1 and s_2, the sets $D(G_L, f, s_1)$ and $D(G_L, f, s_2)$ are distinct; in other words, there must exist $c \in \mathscr{C}_L$ such that $F_{s_1}(c) \neq F_{s_2}(c)$.

The next theorem, first appearing in [20], is included here for completeness and addresses the issue of computing $|U(G_L)|$.

Theorem 2. *For any G_L ECA digraph of length L, $|U(G_L)| = 3^L - 2^{L+1} + 2$.*

Proof. Let *lab* be a labeling of G_L, $((G_L)_R, lab_R)$ be its LRM and notice that:

(i) The only possible forbidden cycles in $((G_L)_R, lab_R)$ have either length two (Fig. 5(a)) or L (examples (b) and (c) in Fig. 5)
(ii) If (G_L, lab) has the pattern P (Fig. 6(a)), then $((G_L)_R, lab_R)$ has the pattern P_R (Fig. 6(b)), which prevents the existence of forbidden cycles of length L in P.
So, the following holds:

(a) If the pattern P exists in (G_L, lab), then there are no forbidden cycles of length L in $((G_L)_R, lab_R)$.
(b) If the pattern P does not exist in (G_L, lab), but (G_L, lab) has at least one cycle of length two with labels $+-$ or $-+$, then there exists a forbidden cycle of length L in $((G_L)_R, lab_R)$.

Thus, $|U(G)|$ may be determined according to Theorem 1 by counting the labeled digraphs that do not create forbidden cycles in their LRMs (the two ones described in the item (i) above):

(1) According to (i), there are three possible ways to label each cycle of length two in order to avoid forbidden cycles of such a length: $++$, $+-$ and $-+$ (but not $--$), hence 3^L possibilities of labeled digraphs without forbidden cycles of length two in their LRMs.
(2) From the 3^L labeled digraphs described in (1), it is necessary to remove those who have a forbidden cycle of length L as follows: choose an orientation for such a cycle that may

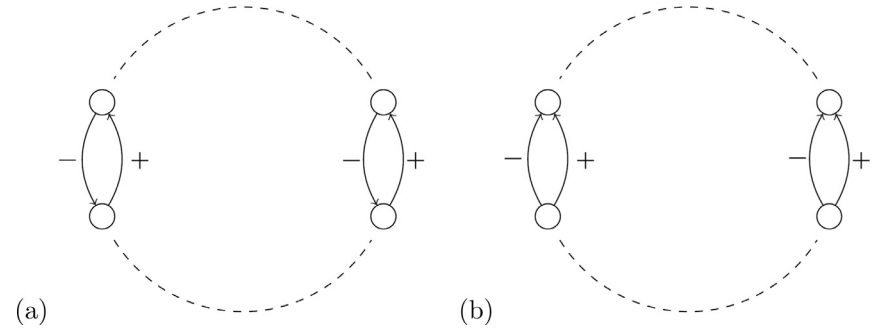

Fig. 6. The pattern P in (G_L, lab) (a) generating pattern P_R in $((G_L)_R, lab_R)$ (b) which prevents the existence of forbidden cycles of length L in the later.

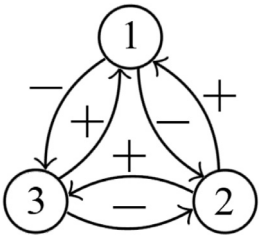

Fig. 7. Example of an update digraph for configurations of length 3. Cell 1 is updated first, then cell 3, and finally cell 2.

appear in the corresponding LRM (either clockwise or counterclockwise). Since all the negative edges must have the same orientation, there are two possible choices for the labels in each one of the L cycles of length two in (G_L, lab): $-+$ or $++$. Hence, there are 2^L digraphs labeled in this way, and only the one with all cycles of the type $++$ is in fact an update digraph. Therefore, there are $2^L - 1$ digraphs with forbidden cycles of length L with the given selected orientation. Repeating this analysis for the other orientation, $2^L - 1$ further digraphs with forbidden cycles of length L are counted. Consequently, there are $2(2^L - 1)$ labels which create digraphs with forbidden cycles of length L in their LRMs. Therefore, $|U(G)| = 3^L - 2(2^L - 1) = 3^L - 2^{L+1} + 2$. □

In the next section we give a characterisation of ECA rules for which $|D(G_L, f)| = |U(G_L)|$.

5. Necessary and sufficient conditions for maximum update schedule sensitivity

In this section, a characterisation of the ECA rules with maximum update schedule sensitivity is given. However, we start by characterising the rules that do not have such a property.

First of all, it is useful to show how an update schedule may be encoded as a sequence of compositions of the local rule. For instance, let us consider the update digraph in Fig. 7.

Cell 1 is updated first, and its neighbours are cell 3 on the left and 2 on the right, neither of which has been updated yet. Hence, its image will be $y_1 = f(c_3, c_1, c_2)$.

Then, cell 3 is to be updated, and its neighbours will be cell 2 to the left, which has not been updated and therefore is in state c_2, and cell 1 on the right, which has been updated and is in state $y_1 = f(c_3, c_1, c_2)$. Hence, the image of cell 3 is given by $y_3 = f(c_2, c_3, y_1) = f\big(c_2, c_3, f(c_3, c_1, c_2)\big)$.

Finally, cell 2 is to be updated and its neighbours have both been already updated, so that its left neighbour, cell 1, is in state $y_1 = f(c_3, c_1, c_2)$ and its right neighbour, cell 3, is in state $y_3 = f\big(c_2, c_3, f(c_3, c_1, c_2)\big)$. Therefore, the image of cell 2 is given by $y_2 = f(y_1, c_2, y_3) = f\Big(f(c_3, c_1, c_2), c_2, f\big(c_2, c_3, f(c_3, c_1, c_2)\big)\Big)$.

That is, the image of the configuration (c_1, c_2, c_3), due to f with update schedule given by the update digraph in Fig. 7, is given by

$$\Bigg(\overbrace{f(c_3, c_1, c_2)}^{y_1}, \overbrace{f\Big(f(c_3, c_1, c_2), c_2, f\big(c_2, c_3, f(c_3, c_1, c_2)\big)\Big)}^{y_2}, \overbrace{f\big(c_2, c_3, f(c_3, c_1, c_2)\big)}^{y_3}\Bigg).$$

In the remainder of the paper we rely on such nested compositions of the local rule in order to compare update schedules. But before we carry on, it is convenient to visualise the idea behind what will be shown afterwards.

Given an update schedule s over configurations of length L and $1 \le i \le L$, we define $H_i^s = \{(j, s(j)) \in \{1, \cdots, L\}^2 : s(j) < s(i)\}$, that is, H_i^s is the set of ordered pairs of cells j updated before cell i in schedule s and their respective update priority $s(j)$.

Let s_1 and s_2 be two update schedules over configurations of length L, a cell $1 \le i \le L$ has *equivalent update histories in* s_1 and s_2 when the following conditions are satisfied:

- The neighbours of cell i which have updated before it are the same in both update schedules, that is, $s_1(i-1) < s_1(i) \Leftrightarrow s_2(i-1) < s_2(i)$ and $s_1(i+1) < s_1(i) \Leftrightarrow s_2(i+1) < s_2(i)$;
- If $s_1(i-1) < s_1(i)$ or $s_2(i-1) < s_2(i)$, then $H_{i-1}^{s_1} = H_{i-1}^{s_2}$;
- If $s_1(i+1) < s_1(i)$ or $s_2(i+1) < s_2(i)$, then $H_{i+1}^{s_1} = H_{i+1}^{s_2}$.

In other words, cell i has equivalent update histories in s_1 and s_2 whenever the neighbours that have been updated before it are the same in both update schedules and the cells updated before such neighbours are the same and have been updated in the same order in s_1 and s_2. Notice that for any length L configuration c, $(F_{s_1}(c))_i = (F_{s_2}(c))_i$ holds if i has equivalent update histories for s_1 and s_2. Such a notion will be addressed later in this section.

Let us suppose there are two distinct non-equivalent update schedules s_1 and s_2 such that, for a particular local rule f, $F_{s_1}(c) = F_{s_2}(c)$ for any configuration c of a given length L. Now, take any fixed $i \in \{1, \cdots, L\}$ for which the update schedules may differ; this index i must be such that, at the moment cell i is to be updated, its neighbours that have already been updated under one schedule are not the same as those that have already been updated under the other schedule. Fig. 8 shows all the possible differences of update order for the cells in the neighbourhood of cell i up to swapping s_1 and s_2. Black bullets indicate that a cell has been updated prior to cell i.

Accordingly, at the top-left of the figure, Case 1 states that cell $i-1$ has been updated before cell i under s_1 only, and that cell $i+1$ has not been updated before i neither under s_1 nor s_2. Let us consider any configuration c with length $L = 5$ and $i = 3$. This

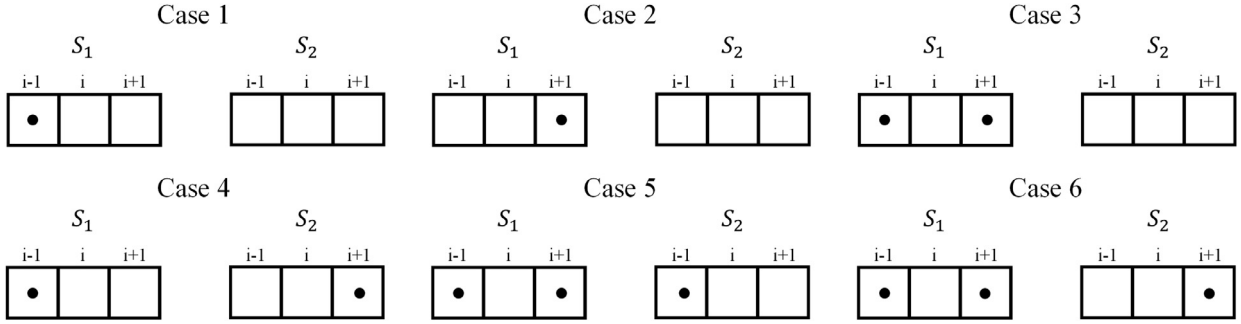

Fig. 8. All possible differences in the update history of cells $i-1$ and $i+1$ for update schedules s_1 and s_2 (up to swapping s_1 and s_2). Black bullets indicate that a cell has been updated before cell i.

means that cell 2 has been updated before cell 3 for update schedule s_1 but not for s_2. However, since cell 2 has been updated, it is necessary to consider whether or not its neighbouring cell 1 has been updated before cell 2. By the same transitive reasoning, we must consider whether cell 5 has been updated before cell 1 (recall that c is periodic). Notice that such a reasoning does not apply to cell 4 since, by hypothesis (Case 1 of Fig. 8), it has not been updated yet.

Hence, for each of the above cases, there is a correspondent equation the local rule f must satisfy for any configuration $c = (c_1, c_2, c_3, c_4, c_5)$, namely:

- If only cell 2 has been updated before cell 3 in s_1 and no cell has been updated before cell 3 in s_2, by the time cell 3 is to be updated, its neighbourhood will be $(f(c_1, c_2, c_3), c_3, c_4)$ in s_1 and (c_2, c_3, c_4) in s_2. Hence, f must satisfy
$$\overbrace{f\big(f(c_1,c_2,c_3),c_3,c_4\big)}^{\left(F_{s_1}(c)\right)_3} = \overbrace{f(c_2,c_3,c_4)}^{\left(F_{s_2}(c)\right)_3};$$
- If cell 2 has been updated before cell 3, and cell 1 has been updated before cell 2 in s_1 and no cell has been updated before cell 3 in s_2, by the time cell 3 is to be updated, its neighbourhood will be $(f(f(c_5, c_1, c_2), c_2, c_3), c_3, c_4)$ in s_1 and (c_2, c_3, c_4) in s_2. Hence, f must satisfy
$$f\big(f\big(f(c_5,c_1,c_2),c_2,c_3\big),c_3,c_4\big) = f(c_2,c_3,c_4);$$
- If cell 2 has been updated before cell 3, cell 1 has been updated before cell 2 and cell 5 has been updated before cell 1 in s_1 and no cell has been updated before cell 3 in s_2, by the time cell 3 is to be updated, its neighbourhood will be $(f(f(f(c_4, c_5, c_1), c_1, c_2), c_2, c_3), c_3, c_4)$ in s_1 and (c_2, c_3, c_4) in s_2. Hence, f must satisfy
$$f\big(f\big(f\big(f(c_4,c_5,c_1),c_1,c_2\big),c_2,c_3\big),c_3,c_4\big) = f(c_2,c_3,c_4).$$

Given a configuration c of length L and $1 \le i \le L$, $f_i^{[m,n]}(c)$ will denote the triple (y_1, y_2, y_3) given by

$$y_1 = f\big(\cdots f\big(f(c_{i-1-m}, c_{i-m}, c_{i-m+1}), c_{i-m+1}, c_{i-m+2}\big), c_{i-1}, c_i\big);$$
$$y_2 = c_i;$$
$$y_3 = f\big(c_i, c_{i+1}, f\big(c_{i+1}, c_{i+2}, f\big(\cdots, c_{i+n-1}, f(c_{i+n-1}, c_{i+n}, c_{i+n+1})\big)\big)\big).$$

That is, $f_i^{[m,n]}(c)$ is a triple representing the neighbourhood of cell i in an update schedule in which cells $(i-m)$ to $(i-1)$ and cells $(i+1)$ to $(i+n)$ have been updated before cell i (in the following order: cell $i-m$ before cell $i-m+1$, cell $i-m+1$ before cell $i-m+2$, ..., cell $i-1$ before cell i, and symmetrically for the other side).

The above three equations for Case 1, $L = 5$ and $i = 3$ are then rewritten as

$$f\big(f_3^{[1,0]}(c)\big) = f(c_2, c_3, c_4)$$
$$f\big(f_3^{[2,0]}(c)\big) = f(c_2, c_3, c_4)$$
$$f\big(f_3^{[3,0]}(c)\big) = f(c_2, c_3, c_4).$$

Notice that if f satisfies one of the above equations, it also satisfies the one that follows. For instance, suppose that f satisfies $f\big(f_3^{[1,0]}(c)\big) = f\big(f_3^{[0,0]}(c)\big)$, that is, f satisfies

$$\begin{cases} f\big(f(0,c_2,c_3),c_3,c_4\big) = f(c_2,c_3,c_4) \\ f\big(f(1,c_2,c_3),c_3,c_4\big) = f(c_2,c_3,c_4). \end{cases}$$

Now, since $f(c_5, c_1, c_2) \in \{0, 1\}$, we have

$$f\big(f\big(f(c_5,c_1,c_2),c_2,c_3\big),c_3,c_4\big) = f\big(f(0,c_2,c_3),c_3,c_4\big) \text{ or}$$
$$f\big(f\big(f(c_5,c_1,c_2),c_2,c_3\big),c_3,c_4\big) = f\big(f(1,c_2,c_3),c_3,c_4\big).$$

hence f satisfies $f\big(f_3^{[2,0]}(c)\big) = f\big(f_3^{[0,0]}(c)\big)$ and so on.

Therefore, in each of the six possible cases shown in Fig. 8 it is only necessary to consider the case where all possible cells have been updated before cell i. But notice that for Cases 3 to 6 there are many possible updates in which all possible cells have been updated before cell i; for instance, considering $L = 5$ and $i = 3$, the following sets of equations are obtained:

- Case 1
$$f\big(f_3^{[3,0]}(c)\big) = f(c_2, c_3, c_4).$$
- Case 2
$$f\big(f_3^{[0,3]}(c)\big) = f(c_2, c_3, c_4).$$
- Case 3
$$f\big(f_3^{[1,4]}(c)\big) = f(c_2, c_3, c_4);$$
$$f\big(f_3^{[2,3]}(c)\big) = f(c_2, c_3, c_4);$$
$$f\big(f_3^{[3,2]}(c)\big) = f(c_2, c_3, c_4);$$
$$f\big(f_3^{[4,1]}(c)\big) = f(c_2, c_3, c_4).$$
- Case 4
$$f\big(f_3^{[4,0]}(c)\big) = f\big(f_3^{[0,1]}(c)\big);$$
$$f\big(f_3^{[3,0]}(c)\big) = f\big(f_3^{[0,2]}(c)\big);$$
$$f\big(f_3^{[2,0]}(c)\big) = f\big(f_3^{[0,3]}(c)\big);$$
$$f\big(f_3^{[1,0]}(c)\big) = f\big(f_3^{[0,4]}(c)\big).$$

- Case 5

$$f\left(f_3^{[1,4]}(c)\right) = f\left(f_3^{[1,0]}(c)\right);$$
$$f\left(f_3^{[2,3]}(c)\right) = f\left(f_3^{[2,0]}(c)\right);$$
$$f\left(f_3^{[3,2]}(c)\right) = f\left(f_3^{[3,0]}(c)\right);$$
$$f\left(f_3^{[4,1]}(c)\right) = f\left(f_3^{[4,0]}(c)\right).$$

- Case 6

$$f\left(f_3^{[1,4]}(c)\right) = f\left(f_3^{[0,4]}(c)\right);$$
$$f\left(f_3^{[2,3]}(c)\right) = f\left(f_3^{[0,3]}(c)\right);$$
$$f\left(f_3^{[3,2]}(c)\right) = f\left(f_3^{[0,2]}(c)\right);$$
$$f\left(f_3^{[4,1]}(c)\right) = f\left(f_3^{[0,1]}(c)\right).$$

The above sets of equations will be denoted by $f(*, c_3, c_4) = f(c_2, c_3, c_4)$, $f(c_2, c_3, *) = f(c_2, c_3, c_4)$, $f(*, c_3, *) = f(c_2, c_3, c_4)$, $f(*, c_3, c_4) = f(c_2, c_3, *)$, $f(*, c_3, *) = f(*, c_3, c_4)$ and $f(*, c_3, *) = f(c_2, c_3, *)$, respectively. More generally, given a configuration c of length L and $1 \le i \le L$, we use the following notation:

- $\left(f(*, c_i, c_{i+1}) = f(c_{i-1}, c_i, c_{i+1})\right) = \{f\left(f_i^{[L-2,0]}(c)\right) = f(c_{i-1}, c_i, c_{i+1})\}$.
- $\left(f(c_{i-1}, c_i, *) = f(c_{i-1}, c_i, c_{i+1})\right) = \{f(f_i^{[0,L-2]}(c)) = f(c_{i-1}, c_i, c_{i+1})\}$.
- $\left(f(*, c_i, *) = f(c_{i-1}, c_i, c_{i+1})\right) = \{f\left(f_i^{[j,L-j]}(c)\right) = f(c_{i-1}, c_i, c_{i+1}) : 1 \le j \le L-1\}$.
- $\left(f(*, c_i, c_{i+1}) = f(c_{i-1}, c_i, *)\right) = \{f\left(f_i^{[L-j,0]}(c)\right) = f\left(f_i^{[0,j]}(c)\right) : 1 \le j \le L-1\}$.
- $\left(f(*, c_i, *) = f(*, c_i, c_{i+1})\right) = \{f\left(f_i^{[j,L-j]}(c)\right) = f\left(f_i^{[j,0]}(c)\right) : 1 \le j \le L-1\}$.
- $\left(f(*, c_i, *) = f(c_{i-1}, c_i, *)\right) = \{f\left(f_i^{[j,L-j]}(c)\right) = f\left(f_i^{[0,j]}(c)\right) : 1 \le j \le L-1\}$.

Notice that each expression with the '*'-symbol is in fact a *set of equations*. A rule is said to satisfy an expression of this form if it satisfies at least one of the equations in the set. The reasons for this notation will be clearer as we proceed.

By exploring the idea depicted in Fig. 8 and briefly discussed above, the following proposition gives necessary conditions for an ECA local rule not to have maximum update schedule sensitivity for configurations of a given length $L \ge 3$.

Proposition 1. *Let f be a local ECA rule. If f does not have maximal update schedule sensitivity for configurations of length $L \ge 3$ then it satisfies at least one of the following conditions for any length L configuration c and any $1 \le i \le L$:*

1. $f(*, c_i, c_{i+1}) = f(c_{i-1}, c_i, c_{i+1})$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$;
2. $f(c_{i-1}, c_i, *) = f(c_{i-1}, c_i, c_{i+1})$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$;
3. $f(*, c_i, f(c_i, c_{i+1}, *)) = f(c_{i-1}, c_i, c_{i+1})$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$;
4. $f(*, c_i, c_{i+1}) = f(c_{i-1}, c_i, *)$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$;
5. $f(*, c_i, *) = f(*, c_i, c_{i+1})$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$;
6. $f(*, c_i, *) = f(c_{i-1}, c_i, *)$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$.

Proof. Since f does not have maximal update schedule sensitivity, there are distinct update schedules s_1 and s_2 such that $F_{s_1}(c) = F_{s_2}(c)$ for any configuration c with length $L \ge 3$.

Let $i \in \{1, \cdots, L\}$ be such that at least one of the cells $i-1$ or or $i+1$ has been updated under s_1 but not under s_2. The following cases describe the possibilities for the neighbourhood of cell i:

1. Cell $i-1$ has been updated under s_1 and cell $i+1$ has not been updated in neither s_1 nor s_2;
2. Cell $i+1$ has been updated under s_1 and cell $i-1$ has not been updated in neither s_1 nor s_2;
3. Both cells $i-1$ and $i+1$ have been updated under s_1 but not in s_2;
4. Cell $i-1$ has been updated only under s_1 and cell $i+1$ has been updated only under s_2;
5. Both cells $i-1$ and $i+1$ have been updated under s_1 and only cell $i-1$ has been updated under s_2;
6. Both cells $i-1$ and $i+1$ have been updated under s_1 and only cell $i+1$ has been updated under s_2.

Since $\left(F_{s_1}(c)\right)_i = \left(F_{s_2}(c)\right)_i$, for all c, for each case we must have:

Case 1: $f(*, c_i, c_{i+1}) = f(c_{i-1}, c_i, c_{i+1})$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$;
Case 2: $f(c_{i-1}, c_i, *) = f(c_{i-1}, c_i, c_{i+1})$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$;
Case 3: $f(*, c_i, *) = f(c_{i-1}, c_i, c_{i+1})$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$;
Case 4: $f(*, c_i, c_{i+1}) = f(c_{i-1}, c_i, *)$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$;
Case 5: $f(*, c_i, *) = f(*, c_i, c_{i+1})$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$;
Case 6: $f(*, c_i, *) = f(c_{i-1}, c_i, *)$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$. □

Remark: In fact, rules satisfying Cases 1, 2 or 4 for $L = 3$ must also satisfy at least one case among cases 3, 5 and 6 due to the fact that the update in any of the cells will affect the other two neighbourhoods. Therefore, for $L = 3$ it suffices to consider Cases 3, 5 and 6.

The following six propositions will give the reciprocal of Proposition 1. The general idea is to provide a pair of update schedules for rules complying with at least one of the cases described in Proposition 1, for which the images of a configuration by the rule under both update schedules are the same. In each case we highlight the differences in the update digraphs of each update schedule in order to show that they are not equivalent to each other.

Since each expression with a '*' symbol represents a set of equations, with a certain depth, a family of pairs of update schedules are provided in each case, instead of only a single pair. Basically, each pair of updates of the family satisfies one of the possible equations emerging out of the depth (indexed by parameter j) of the expression implicit to the '*' notation.

Proposition 2. *Let f be a local ECA rule. If f is such that*

$$f(*, c_i, c_{i+1}) = f(c_{i-1}, c_i, c_{i+1})$$

for any configuration c of a given length $L > 3$, for any $1 \le i \le L$, then it does not have maximal update schedule sensitivity for configurations of length L.

Proof. Let c be a configuration of length $L > 3$. Since f satisfies $f(*, c_i, c_{i+1}) = f(c_{i-1}, c_i, c_{i+1})$, in particular, it satisfies $f\left(f^{[L-2,0]}(c_L, c_1, c_2)\right) = f(c_L, c_1, c_2)$.

Consider the update schedules s_1 and s_2 given by:

$$s_1(i) = \begin{cases} i-2, & \text{if } 3 \le i \le L \\ L-1, & \text{otherwise.} \end{cases} \qquad s_2(i) = \begin{cases} s_1(i), & \text{if } 3 \le i \le L-1 \\ L-2, & \text{otherwise.} \end{cases}$$

The update schedules are not equivalent, since their update digraphs differ (see Fig. 9).

Basically, the idea is to provide a pair of update schedules such that the update histories of every cell except one is the same in both schedules. In the present case, cells 2 to L have the same update histories, but not for cell 1: in s_1 only its left neighbour has been updated, while in s_2 neither neighbours have been updated. In other words, $\left(F_{s_1}(c)\right)_i = \left(F_{s_2}(c)\right)_i$ for all $2 \le i \le L$ because the update histories for such cells are the same in both update schedules. It remains to show that $\left(F_{s_1}(c)\right)_1 = \left(F_{s_2}(c)\right)_1$.

By the definition of s_1 and s_2, we have $\left(F_{s_1}(c)\right)_1 = f\left(f^{[L-2,0]}(c_L, c_1, c_2)\right)$ and $\left(F_{s_2}(c)\right)_1 = f(c_L, c_1, c_2)$. Since f

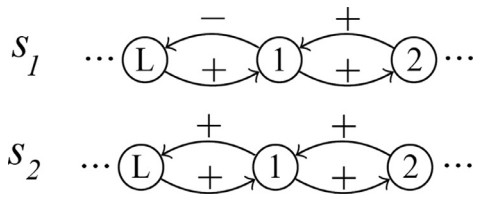

Fig. 9. Difference between the update digraphs of s_1 and s_2 in Proposition 2.

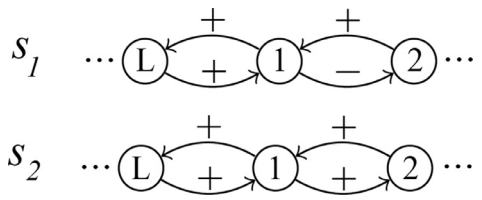

Fig. 10. Difference between the update digraphs of s_1 and s_2 in Proposition 3.

satisfies $f\left(f^{[L-2,0]}(c_L, c_1, c_2)\right) = f(c_L, c_1, c_2)$, it follows that $\left(F_{s_1}(c)\right)_1 = \left(F_{s_2}(c)\right)_1$.

Then $F_{s_1}(c) = F_{s_2}(c)$ and f does not have maximal update schedule sensitivity for configurations of length L. □

The proofs of Propositions 3 to 7 are essentially the same as the one provided above for Proposition 2. In order to avoid unnecessary repetitions, we omitted the common details, providing only the pairs of updates yielding the same image.

Proposition 3. *Let f be a local ECA rule. If f is such that*

$$f(c_{i-1}, c_i, *) = f(c_{i-1}, c_i, c_{i+1})$$

for any configuration c of a given length $L > 3$, for any $1 \le i \le L$, then it does not have maximal update schedule sensitivity for configurations of length L.

Proof. Take update schedules s_1 and s_2 given by:

$$s_1(i) = \begin{cases} L-i, & \text{if } 2 \le i \le L-1 \\ L-1, & \text{otherwise.} \end{cases} \qquad s_2(i) = \begin{cases} s_1(i), & \text{if } 3 \le i \le L-1 \\ L-2, & \text{otherwise.} \end{cases}$$

The update schedules are not equivalent, since their update digraphs differ (see Fig. 10) and $F_{s_1}(c) = F_{s_2}(c)$. □

Proposition 4. *Let f be a local ECA rule. If f is such that*

$$f(*, c_i, *) = f(c_{i-1}, c_i, c_{i+1})$$

for any configuration c of a given length $L \ge 3$, for any $1 \le i \le L$, then it does not have maximal update schedule sensitivity for configurations of length L.

Proof. If $L = 3$, take the following pair of update schedules:

$$s_1(i) = \begin{cases} 1, & \text{if } i \in \{2, 3\} \\ 2, & \text{otherwise.} \end{cases} \qquad s_2(i) = 1, \text{ for all } i \in \{1, 2, 3\}$$

If $L > 3$, take the following pair of update schedules:

$$s_1(i) = \begin{cases} j-i+1, & \text{if } 2 \le i \le j \\ i-1, & \text{if } j < i \le L \\ L, & \text{otherwise.} \end{cases} \qquad s_2(i) = \begin{cases} s_1^j(i), & \text{if } 3 \le i \le L-1 \\ L-3, & \text{otherwise.} \end{cases}$$

In any case, the update schedules are not equivalent, since their update digraphs differ (see Fig. 11) and $F_{s_1}(c) = F_{s_2}(c)$. □

Proposition 5. *Let f be a local ECA rule. If f is such that*

$$f(*, c_i, c_{i+1}) = f(c_{i-1}, c_i, *)$$

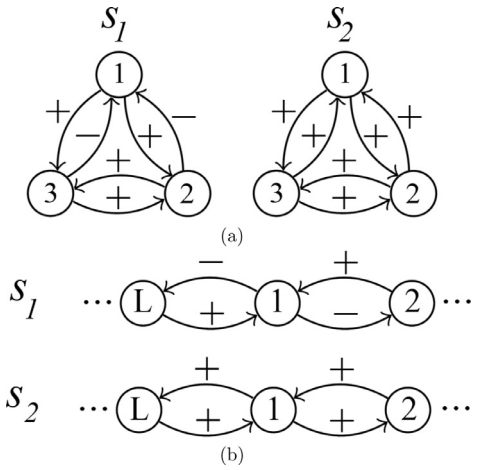

Fig. 11. Difference between the update digraphs of s_1 and s_2 in Proposition 4 for configurations of (a) length $L = 3$ and of (b) length $L > 3$.

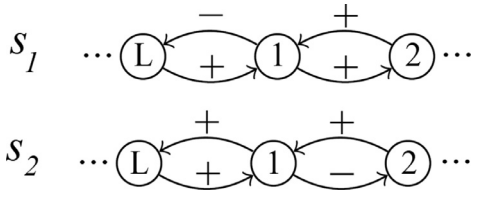

Fig. 12. Difference between the update digraphs of s_1 and s_2 in Proposition 5 for $L > 3$.

for any configuration c of a given length $L > 3$, then it does not have maximal update schedule sensitivity for configurations of length L.

Proof. Consider the following pair of update schedules:

$$s_1(i) = \begin{cases} j-i+1, & \text{if } 3 \le i \le j+2 \\ i-2, & \text{if } j+2 < i \le L \\ L-1, & \text{otherwise.} \end{cases}$$

$$s_2(i) = \begin{cases} j-i+1, & \text{if } 2 \le i \le j+2 \\ i-2, & \text{if } j+2 < i \le L-1 \\ L-1, & \text{otherwise.} \end{cases}$$

The update schedules are not equivalent, since their update digraphs differ (see Fig. 12) and $F_{s_1}(c) = F_{s_2}(c)$. □

Proposition 6. *Let f be a local ECA rule. If f is such that*

$$f(*, c_i, *) = f(*, c_i, c_{i+1})$$

for any configuration c of a given length $L \ge 3$, then it does not have maximal update schedule sensitivity for configurations of length L.

Proof. If $L = 3$, take the following pair of update schedules:

$$s_1(i) = \begin{cases} 1, & \text{if } i = 3 \\ 2, & \text{if } i = 2 \\ 3, & \text{otherwise.} \end{cases} \qquad s_1(i) = \begin{cases} 1, & \text{if } i = 3 \\ 2, & \text{otherwise.} \end{cases}$$

If $L > 3$, take instead the following pair of update schedules:

$$s_1(i) = \begin{cases} i-L+j, & \text{if } L-j < i \le L \\ j+L-i, & \text{if } 2 \le i \le L-j \\ L, & \text{if } i = 1. \end{cases} \qquad s_2(i) = \begin{cases} s_1(i), & \text{if } 3 \le i \le L \\ L-1, & \text{otherwise.} \end{cases}$$

In any case, the update schedules are not equivalent, since their update digraphs differ (see Fig. 13), and $F_{s_1}(c) = F_{s_2}(c)$. □

Proposition 7. *Let f be a local ECA rule. If f is such that*

$$f(*, c_i, *) = f(c_{i-1}, c_i, *)$$

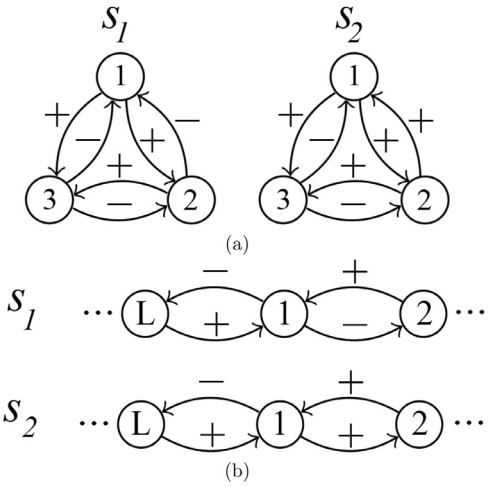

Fig. 13. Difference between the update digraphs of s_1 and s_2 in Proposition 6 for configurations of (a) length $L = 3$ and of (b) length $L > 3$.

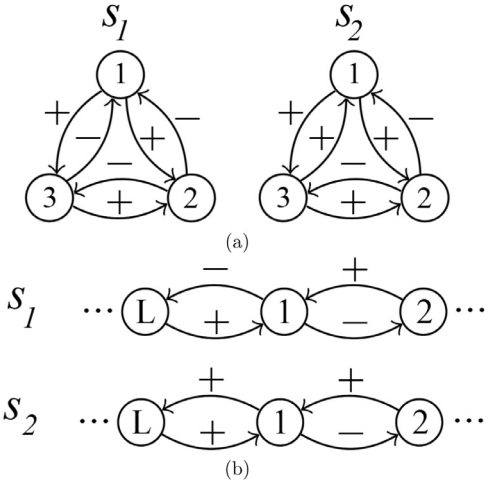

Fig. 14. Difference between the update digraphs of s_1 and s_2 in Proposition 7 for configurations of (a) length $L = 3$ and of (b) length $L > 3$.

for any configuration c of a given length $L \geq 3$, then it does not have maximal update schedule sensitivity for configurations of length L.

Proof. If $L = 3$, consider the following pair of update schedules:

$$s_1(i) = \begin{cases} 1, & \text{if } i = 2 \\ 2, & \text{if } i = 3 \\ 3, & \text{otherwise.} \end{cases} \qquad s_2(i) = \begin{cases} 1, & \text{if } i = 2 \\ 2, & \text{otherwise.} \end{cases}$$

If $L > 3$, take instead s_1 and s_2 as follows:

$$s_1(i) = \begin{cases} i-L+j, & \text{if } L-j < i \leq L \\ j+L-i, & \text{if } 2 \leq i \leq L-j \\ L, & \text{if } i = 1. \end{cases} \qquad s_2(i) = \begin{cases} s_1(i), & \text{if } 2 \leq i \leq L-1 \\ L-1, & \text{otherwise.} \end{cases}$$

In any case, the update schedules are not equivalent, since their update digraphs differ (see Fig. 14), and $F_{s_1}(c) = F_{s_2}(c)$. □

Theorem 3. *Let f be a local ECA rule. It does not have maximal update schedule sensitivity for configurations of a given length $L \geq 3$ if, and only if, it satisfies at least one of the following conditions for any length L configuration c:*

1. $f(*, c_i, c_{i+1}) = f(c_{i-1}, c_i, c_{i+1})$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$
2. $f(c_{i-1}, c_i, *) = f(c_{i-1}, c_i, c_{i+1})$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$
3. $f(*, c_i, *) = f(c_{i-1}, c_i, c_{i+1})$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$
4. $f(*, c_i, c_{i+1}) = f(c_{i-1}, c_i, *)$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$
5. $f(*, c_i, *) = f(*, c_i, c_{i+1})$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$
6. $f(*, c_i, *) = f(c_{i-1}, c_i, *)$, for all $(c_{i-1}, c_i, c_{i+1}) \in \{0,1\}^3$

Proof. This is a direct result from Proposition 1 and Propositions 2 to 7. □

The above theorem gives us a partition of the ECA rule space relative to maximum sensitivity to update schedules for configurations of any given length $L \geq 3$. Table 1 shows dynamically independent rules according to their classification with this scheme for $L = 3$ up to $L = 9$.

In all cases, most of the rules present maximal update schedule sensitivity. For configurations of length $L \in \{7, 8, 9\}$, the rules with nonmaximal update schedule sensitivity are exactly the same. One may conjecture that such a property will hold for $L > 9$, since the equations an ECA must satisfy to possess maximal update schedule sensitivity are really equations on its local rule, which depends on the cell and its nearest-neighbours and the list of rules displaying such a property remains unchanged as cells are added from $L = 7$ to $L = 9$.

In fact, rules changing classes from $L = 3$ to $L = 6$ seem to do so due to particular characteristics of them when applied to sufficiently small neighbourhoods with wrap-around effects arising from the periodicity of the configurations.

In other words, the neighbourhoods of cells in small configurations are highly coupled, with any change in one cell caused by asynchronism spreads, directly or indirectly, to the neighbourhoods of most of the remaining cells, making their transitions to be highly correlated. This effect is mitigated by the length of the configurations and seem to be kept constant for $L \geq 7$.

In order to look at the correlations between the transitions in smaller configurations, take for instance ECA local rule 156 ($f = f_{156}$). Such a rule satisfies $f_1^{[2,1]}(c) = f_1^{[0,0]}(c)$ for any configuration c of length 3 but there are configurations $\hat{c}$ of length 4 such that rule 156 does not satisfy the analogous condition $f_1^{[2,2]}(\hat{c}) = f_1^{[0,0]}(\hat{c})$. More precisely, the above equations can be written, respectively, as

$$\begin{cases} f(f(f(c_1, c_2, c_3), c_3, c_1), c_1, f(c_1, c_2, c_3)) = f(c_3, c_1, c_2) \\ f(f(f(\hat{c}_2, \hat{c}_3, \hat{c}_4), \hat{c}_4, \hat{c}_1), \hat{c}_1, f(\hat{c}_1, \hat{c}_2, f(\hat{c}_2, \hat{c}_3, \hat{c}_4))) \\ \quad = f(\hat{c}_4, \hat{c}_1, \hat{c}_2) \end{cases}$$

There are one reoccurring transitions in both cases: $f(c_1, c_2, c_3)$ in the first equation and $f(\hat{c}_2, \hat{c}_3, \hat{c}_4)$ in the second. However, such reoccurring transitions affect the equations differently. Indeed, in order to ease the visualization, let $y_2 = f(c_1, c_2, c_3)$ and $\hat{y}_2 = f(\hat{c}_2, \hat{c}_3, \hat{c}_4)$. Now the equations become

$$\begin{cases} f(f(y_2, c_3, c_1), c_1, y_2) = f(c_3, c_1, c_2) \\ f(f(\hat{y}_2, \hat{c}_4, \hat{c}_1), \hat{c}_1, f(\hat{c}_1, \hat{c}_2, \hat{y}_2)) = f(\hat{c}_4, \hat{c}_1, \hat{c}_2) \end{cases}$$

The second equation cannot be reduced to the first for configurations $\hat{c}$ such that $f(\hat{c}_1, \hat{c}_2, \hat{y}_2) \neq \hat{y}_2$, showing that it is more strict than the first one.

Table 1
Partition of the ECA rule space according to maximum update schedule sensitivity for configurations of length L from 3 to 9. Rules that have changed class from a given length to the subsequent length are highlighted in bold face.

Length	Nonmaximal update schedule sensitivity	Maximal update schedule sensitivity
3	0, 3, 8, 12, 13, 14, 15, 18, 25, 28, 29, 32, 34, 44, 45, 51, 60, 62, 72, 76, 77, 78, 90, 128, 136, 140, 142, 156, 160, 162, 164, 170, 200, 204 and 232	1, 2, 4, 5, 6, 7, 9, 10, 11, 19, 22, 23, 24, 26, 27, 30, 33, 35, 36, 37, 38, 40, 41, 42, 43, 46, 50, 54, 56, 57, 58, 73, 74, 94, 104, 105, 106, 108, 110, 122, 126, 130, 132, 134, 138, 146, 150,152, 154, 168, 172, 178 and 184
	(35 classes)	(53 classes)
4	0, 3, 8, 11, 12, 13, 15, 25, 28, 32, **33**, 34, 44, 51, 60, 76, 77, **104, 105, 122**, 128, 136, 140, **150**, 160, 162, 170, 200, 204 and 232	1, 2, 4, 5, 6, 7, 9, 10, 14, **18**, 19, 22, 23, 24, 26, 27, **29**, 30, 35, 36, 37, 38, 40, 41, 42, 43, **45**, 46, 50, 54, 56, 57, 58, **62, 72**, 73, 74, **78, 90**, 94, 106, 108, 110, 126, 130, 132, 134, 138, **142**, 146, 152, 154, **156, 164**, 168, 172, 178 and 184
	(30 classes)	(58 classes)
5	0, 3, 8, 12, 15, 28, 32, 34, 44, 51, 60, 128, 136, 140, 160, 162, 170, 200, 204 and 232	1, 2, 4, 5, 6, 7, 9, 10, **11, 13**, 14, 18, 19, 22, 23, 24, **25**, 26, 27, 29, 30, **33**, 35, 36, 37, 38, 40, 41, 42, 43, 45, 46, 50, 54, 56, 57, 58, 62, 72, 73, 74, **76, 77**, 78, 90, 94, **104, 105**, 106, 108, 110, **122**, 126, 130, 132, 134, 138, 142, 146, **150**, 152, 154, 156, 164, 168, 172, 178 and 184
	(20 classes)	(68 classes)
6	0, 3, 8, 12, **13**, 15, 28, 32, 34, 44, 51, 60, **90**, 128, 136, 140, 160, 162, 170, 200 and 204	1, 2, 4, 5, 6, 7, 9, 10, 11, 14, 18, 19, 22, 23, 24, 25, 26, 27, 29, 30, 33, 35, 36, 37, 38, 40, 41, 42, 43, 45, 46, 50, 54, 56, 57, 58, 62, 72, 73, 74, 76, 77, 78, 94, 104, 105, 106, 108, 110, 122, 126, 130, 132, 134, 138, 142, 146, 150, 152, 154, 156, 164, 168, 172, 178, 184 and **232**
	(21 classes)	(67 classes)
7 to 9	0, 3, 8, 12, 15, 28, 32, 34, 44, 51, 60, 128, 136, 140, 160, 162, 170, 200 and 204	1, 2, 4, 5, 6, 7, 9, 10, 11, **13**, 14, 18, 19, 22, 23, 24, 25, 26, 27, 29, 30, 33, 35, 36, 37, 38, 40, 41, 42, 43, 45, 46, 50, 54, 56, 57, 58, 62, 72, 73, 74, 76, 77, 78, **90**, 94, 104, 105, 106, 108, 110, 122, 126, 130, 132, 134, 138, 142, 146, 150, 152, 154, 156, 164, 168, 172, 178, 184 and 232
	(19 classes)	(69 classes)

6. Concluding remarks

This work tackled the notion of maximum update schedule sensitivity defined in terms of the distinct one-step trajectories a CA rule generates when subjected to update schedules that are almost identical. In spite of this one-step maximum sensitivity not being sufficient, it is a necessary condition in order for a CA rule to present distinct time-evolutions from the same initial configuration iterated under update schedules with small differences. In other words, the (one-step) maximum update schedule sensitivity may be regarded as a condition a rule must satisfy in order to be sensitive in the long term to perturbations in a given update schedule.

A characterisation of the ECA rules with maximum update schedule sensitivity was given. The proof yielding such a characterisation provides not only the conditions a rule must satisfy in order to present this property, but also pairs of update schedules that make rules with nonmaximal update schedule sensitivity to generate the same image for any given configuration of a fixed length L.

The ECA rule space was then partitioned according to update schedule sensitivity for configurations of length $3 \leq L \leq 9$. For configurations of length 7, 8 and 9 the dynamical classes of rules with nonmaximal sensitivity to update schedules remained the same, suggesting that it may be the case that it suffices to check whether or not a particular rule has maximal sensitivity to update schedules for configurations up to length 9. Also, the characterisation shown rule may be generalised for k-ary rule spaces of larger radius by exploring the generalisation of Cases 1 to 6 shown in the previous section.

The group with maximal schedule sensitivity is composed by rules with diverse complexity for synchronous update. Rule 1, for instance is periodic according to the classification scheme in [21], while Rule 18 is chaotic and Rule 110 is complex according to the same classification. This suggests that it may be convenient to consider a finer notion of update schedule sensitivity, such as one considering the number of configurations yielding the same image.

Also, intuitively, a more sensitive rule has more chance of producing distinct kinds of behaviour for distinct update schedules. Hence, one may study how richer the dynamical behaviour of a given rule may become by using different update schedules and how the typical time evolution of a rule would be affected by small changes in the update schedule.

Finally, by knowing which rules have more sensitivity to changes in update schedules and, therefore, a larger set of distinct possible behaviours, the choice of rules to solve particular decision problems – such as density determination, as hinted at earlier – could be made among such rules.

Declarations of interest

none.

Acknowledgements

This work was partially supported by FONDECYT Iniciación 11150827 (M.M-M.), P.P.B.O. thanks CNPq and K.P. thanks ECOS-C16E01, PACA project FRI-2015_01134 and PEPS JCJC INS2I project CGETA.

We also thank one of the anonymous reviewers for extremely detailed comments and suggestions to our originally submitted manuscript.

References

[1] Hoekstra A, Kroc J, Sloot P. Introduction to modeling of complex systems using cellular automata. Simul Complex Syst Cellular Automata 2010:1–16.
[2] Mikler AR, Venkatachalam S, Abbas K. Modeling infectious diseases using global stochastic cellular automata. J Biol Syst 2005;13(04):421–39.
[3] Aburas MM, Ho YM, Ramli MF, Ash'aari ZH. The simulation and prediction of spatio-temporal urban growth trends using cellular automata models: a review. Int J Appl Earth Obs Geoinf 2016;52:380–9.
[4] Wolf-Gladrow DA. Lattice-Gas Cellular Automata and Lattice Boltzmann Models. Springer; 2000.
[5] Gardner M. Mathematical games: the fantastic combinations of john conway's new solitaire game "life". Sci Am 1970;223(4):120–3.
[6] Cook M. Universality in elementary cellular automata. Complex Syst 2004;15(1):1–40.
[7] Fatès N. A guided tour of asynchronous cellular automata. J Cellular Automata 2014;9(5–6):387–416.
[8] Messinger SM, Mott KA, Peak D. Task-performing dynamics in irregular, biomimetic networks. Complexity 2007;12(6):14–21.
[9] Bersini H, Detours V. Asynchrony induces stability in cellular automata based models. In: Artificial Life IV. MIT Press, MA; 1994. p. 382–7.
[10] Fatès N, Morvan M. An experimental study of robustness to asynchronism for elementary cellular automata. Complex Systems 2005;16:1–27.
[11] Vielhaber M. Computation of functions on n bits by asynchronous clocking of cellular automata. Nat Comput 2013;12(3):307–22.
[12] Aracena J, Demongeot J, Fanchon E, Montalva M. On the number of update digraphs and its relation with the feedback arc sets and tournaments. Discrete Appl Math 2013;161(10):1345–55.
[13] Aracena J. Maximum number of fixed points in regulatory boolean networks. Bull Math Biol 2008;70(5):1398.
[14] Richard A. Fixed points and connections between positive and negative cycles in boolean networks. Discrete Appl Math 2018.
[15] Remy E, Ruet P, Thieffry D. Graphic requirements for multistability and attractive cycles in a boolean dynamical framework. Adv Appl Math 2008;41(3):335–50.
[16] de Oliveira PPB. On density determination with cellular automata: results, constructions and directions. J Cellular Automata 2014;9(5–6):357–85.
[17] Wolfram S. A new kind of science. Wolfram Media; 2002.
[18] Aracena J, Fanchon E, Montalva M, Noual M. Combinatorics on update digraphs in boolean networks. Discrete Appl Math 2011;159(6):401–9.
[19] Aracena J, Goles E, Moreira A, Salinas L. On the robustness of update schedules in boolean networks. BioSystems 2009;97:1–8.
[20] Montalva M, Perrot K, de Oliveira P, Ruivo E. Sensitivity to synchronism in some boolean automata networks. In: 23rd International Workshop on Cellular Automata and Discrete Complex Systems – AUTOMATA 2017 Exploratory Papers proceedings; 2017. p. 69–76.
[21] Li W, Packard N. The structure of the elementary cellular automata rule space. Structure 1990;4:281–97.