Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2015, Article ID 594294, 14 pages
http://dx.doi.org/10.1155/2015/594294

Review Article

Parallel Dynamical Systems over Graphs and Related Topics: A Survey

Juan A. Aledo, Silvia Martinez, and Jose C. Valverde

Department of Mathematics, University of Castilla-La Mancha, 02071 Albacete, Spain

Correspondence should be addressed to Jose C. Valverde; jose.valverde@uclm.es

Received 10 October 2014; Accepted 10 December 2014

Academic Editor: Giuseppe Marino

In discrete processes, as computational or genetic ones, there are many entities and each entity has a state at a given time. The update of states of the entities constitutes an evolution in time of the system, that is, a discrete dynamical system. The relations among entities are usually represented by a graph. The update of the states is determined by the relations of the entities and some local functions which together constitute (global) evolution operator of the dynamical system. If the states of the entities are updated in a synchronous manner, the system is called a *parallel dynamical system*. This paper is devoted to review the main results on the dynamical behavior of parallel dynamical systems over graphs which constitute a generic tool for modeling discrete processes.

1. Introduction

Modeling discrete processes is one of the most important tasks in modern mathematics. In fact, several mathematical concepts have become fundamental in order to establish discrete mathematical models for several phenomena coming from science and engineering. Some of them are *Boolean algebras* and *functions* [1, 2], together with other topics as *graphs* [1, 2] and *discrete dynamical systems* [3–6].

Boolean algebras give symbolic form to Aristotle's system of logic. In the second half of the nineteenth century, the English mathematician George Boole (1815–1864) defined an algebraic structure which encodes several rules of relationship between mathematical quantities limited to two possible values: true or false, mathematically modeled by 1 or 0. This algebraic structure is currently known as Boolean algebra. The results of the study were published in a survey [7] entitled "An Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities" in 1854. Nevertheless, after almost a century, Claude E. Shannon was the first one who established how Boolean algebras could be applied to on-off circuits, where the presence of signal is characterized by 1 and the absence by 0. In the forties, his master's thesis [8] entitled "A Symbolic Analysis of Relay and Switching Circuits" made Boole's theoretical work become a fundamental mathematical tool for designing and analyzing digital circuits.

Graphs appear in many different fields of science and engineering as a tool to represent the relations among the elements of a system [9–15]. They constitute a powerful tool to perceive more clearly problems and to face up to them, especially when there is a finite number of elements to deal with. This occurs in discrete processes and so they are basic for the mathematical formalization of many of these processes.

The notion of *dynamical system* is the mathematical formalization of the general scientific concept of deterministic process [5, 6]. It involves a set of possible states of the process, which is named *state space* or *phase space*, and a *law of the evolution* or *evolution operator* in *time*. Thus, this notion is fundamental in order to model any phenomenon whose future asymptotic state needs to be known.

These three concepts are mixed properly to construct a mathematical model named *parallel dynamical system* (PDS) that allows us to formalize and analyze the dynamical behavior of discrete processes.

This mathematical model constitutes a generalization of other relevant ones which appeared previously in the literature, as *cellular automata* (CA) [16–25] or *Boolean networks* (BN) [14, 26–29]. (The abbreviations PDS, SDS, CA, and BN will be written for the singular and plural forms of

the corresponding terms, since it seems better from an aesthetic point of view).

A CA has been traditionally conceived as a grid of cells, where each cell has a state belonging to a finite set (usually $\{0, 1\}$), such that all the cells evolve synchronously in discrete time steps. The updating of the state of every cell is realized according to a common local function affecting only the neighbors of such a cell. Thus, if x_i^t is the state value of a cell i at the time t, the cell value is updating by applying a local function on the cells in the neighborhood of the cell i.

CA has revealed as a suitable mathematical model to capture the essential features of digital computers: synchronicity, regular distribution, and locality of iterations. Although they were introduced for the first time in the works of Ulam and von Neumann [30], they became of public interest in 1970, when Martin Gardner published [31] with an explanation of John Conway's "Game of Life" in Scientific American (see also [32]).

In [21], Wolfram analyzed a set of CA and showed that, despite their simple construction, some of them are capable of a complex behavior. Later, in [22], based on a deeper research, he suggested that many one-dimensional cellular automata fall into four basic behavior classes: three of them exhibiting a similar behavior to fixed points, periodic orbits, and chaotic attractors, and the fourth one such that its asymptotic properties are undecidable. Wolfram published other important papers which are compiled in [25].

A BN of size n and connectivity k consists of n interconnected vertices, each one having k inputs. The update of the state of any vertex is determined by the (directed) dependency relations and local rules which are given by Boolean functions. Hence, BN are a generalization of (finite) Boolean CA.

BN appeared, almost at the same time, in applied models created for the simulation of aspects of the behavior of biological systems. This occurred in [26], where Kauffman constructed molecular automata for modeling a gene as a binary (on-off) device and studied the behavior of large, randomly constructed nets of these binary genes (see also [27]). For this reason, BN are also known as *Kauffman (net) models*. The results by Kauffman suggest that if each gene is directly affected by two or three genes then the system behaves with great order and stability and presents cycles.

In the last two decades, many works have focused their attention on mathematical modeling of several computer processes. The first one of a series of these works was [33], which constituted an important step in the development of mathematical foundations for the theory of computation. In this work, sequentially updated cellular automata (SCA) over arbitrary graphs are employed as a paradigmatic framework. This first work was followed by [34–36], where the authors developed this theory, analyzing the asymptotic behavior of such mathematical models. The formal definition of PDS appeared for the first time in [9] and it constitutes a fundamental issue for the posterior development of the results in this research line.

Computer processes involve generation of dynamics by iterating local mappings. In fact, a computer simulation is a method for the composition of iterated mappings, typically on local dependency regions [33]. It means that the mappings have to be updated in a specific manner, that is, an *update schedule*. Update scheduling is also a commonly studied aspect in discrete event simulations [37–39]. Mathematical modeling of computer processes has resulted very useful in order to predict what can occur after executing these processes.

In the computational context, it is common to rename the local mappings as *entities* (cells in the language of cellular automata theory), which are the lowest level of aggregation of the system. In computer processes, there are many entities and each entity has a state at a given time (see [33–35]). The update of states of the entities constitutes an evolution in time of the system, that is, a discrete dynamical system (see [9, 40]).

In this sense, each entity i of the system could be activated or deactivated. Thus, it is natural to consider that its state $x_i \in \{0, 1\}$. The evolution or update of the system is implemented by *local functions* which are restrictions of a global function or together constitute this global one. That is, for updating the state of any entity, the corresponding local function acts only on the state of that entity itself and the states of the entities related to it. The relations among entities are usually represented by a graph which is called the *dependency graph* of the system.

Thus, the update of the states is determined by the dependency graph relations of the entities and the local functions which together constitute the (global) *evolution operator* of the dynamical system. If the states of the entities are updated in a parallel (or synchronous) manner, the system is called a *parallel dynamical system* [9, 41–43], while if they are updated in a sequential (or asynchronous) order, the system is named *sequential dynamical system* (SDS) [9, 33–36, 40, 44–46].

CA, when finite, can be considered as a special kind of PDS by considering cells as entities. On the other hand, BN are a generalization of (finite) Boolean CA but, at the same time, a particular case of PDS by considering nodes as entities. One of the main differences with CA is that, in BN, the state of each node is not affected necessarily by its neighbors but potentially by whichever node in the network. Thus, the uniform structure of neighborhood in CA disappears. However, some homogeneity remains, since each node is affected by k connections with other (or the same) entities. This homogeneity makes BN a particular case of PDS, since in PDS connections can be totally arbitrary.

Since CA and BN are special cases of PDS, their applications can be assumed as applications of PDS. Some applications of CA and BN that appear in the literature are enumerated below in order to give an idea of the great interest of this tool for modeling discrete processes from science and engineering.

As said before, Boolean networks appear for the first time in [26], where Kauffman constructed molecular automata for modeling a gene as a binary (on-off) device (see also [27]). Up to date, these results continue being developed as can be seen in [11], where they consider asynchronous stochastic update, [14], where probability is introduced in the directed dependency graph which they call *random Boolean*

network, or [47], where a Kauffman net model shows that eukaryotic cells are dynamically ordered or critical but not chaotic. Since Kauffman nets can be seen as particular cases of directed dependency graphs that are considered here, the results shown in this review article could be useful in the development of some aspects of biological systems and biochemical control networks.

In [48] the authors show different environments where CA can be used to model several biological patterns. Most recently, Boolean networks have been applied for modeling epidemics in [15].

In the eighties, two important works appeared relating CA and ecological modeling [18, 49]. In fact, [49] presents CA as a paradigm for ecological modeling. Later, in [50] CA were used to simplify spatial complexity in the geometry of ecological interactions. Finally, in [51] the authors studied the links between CA and population dynamics.

More recently, two works [10, 12] have been dedicated to the study of the applications of CA in cryptography.

On the other hand, in [52] a parallel discrete dynamical model with three evolution local functions is used in order to analyze the structure of a partially ordered set (poset) of signed integer partitions whose main properties are actually not known. The authors affirm that this model is related to the study of some extremal combinatorial sum problems. Besides, in [53], a class of lattices and Boolean functions are employed to study the truth of a mathematical conjecture. Other related interesting works in this research line are [54–57].

In the book [58], the use of CA for modeling some problems of physics is studied in detail, while in the book [59], the authors investigate the use of CA in modeling chemical phenomena.

Also in sociology, there exist some works where CA are used for modeling the corresponding systems (see, for instance, [60, 61]). Besides, in the literature one can find other works where this kind of systems are employed for other applications as the simulation of two-lane traffic [62] or the generation of rhythmic structure in music [63].

According to [6], one of the main goals in the study of a dynamical system is to give a complete characterization of its orbit structure. Actually, this is the main purpose of the review paper in relation with parallel dynamical systems over graphs: to show as much information about the orbit structure as possible, based on the properties of the dependency graph and the local Boolean functions which constitute the evolution law of the system.

In this particular case, as the state space of the system is finite, every orbit is periodic or eventually periodic. Note that, for a system with n entities, the number of possible states is equal to 2^n. Thus, for example, if $n = 20$, then the number of possible (initial) states of the system is greater than a million. Therefore, computational studies become inefficient when n is big enough and can only analyze small systems or a low number of initial states.

In view of that, the unique way to give general results for these systems is to find out their properties analytically, as done in [41–43, 64, 65] in order to describe the different coexistent periodic orbits of PDS. Following Derrida and Pomeau [66], the most interesting questions about these systems concerning their orbit structures are the following.

(1) What is the length of the limit cycles?

(2) What is the number of different limit cycles?

(3) If one considers two different initial states, when do they arrive in the same limit cycle?

Actually, these are the main questions reviewed in this paper for PDS. More specifically, this paper reviews the full characterization of the orbit structure of PDS over undirected graphs covering all the casuistry, since it shows the behavior of any parallel dynamical system regardless the type of graph, the number of entities, and the relationships among them. Also, essential properties of the orbit structure of PDS over directed dependency graphs are revised. Besides, two extensions of the concept of PDS are shown.

The paper is organized as follows. In the next section we present theoretical foundations of PDS and compare them with CA and BN in detail. In Section 3, the most important results on the dynamics of PDS are reviewed. Section 4 is devoted to reviewing two extensions of this mathematical model. A linearization algorithm for the computation of orbits in PDS is addressed in Section 5. The paper finishes by setting out several interesting future research directions for the development of these kinds of models.

2. Theoretical Foundations of PDS and Comparative Study with Related Topics

2.1. Theoretical Foundations

2.1.1. Dependency Graphs. In computer processes, there are many entities and each entity has a state at a given time (see [33–36]). Entities are related and they get information from the entities in their neighborhood. Usually, in order to get a graphical idea of the situation, every entity is represented by a vertex of an undirected graph and two vertices are adjacent if their states influence each other in the update of the system. The undirected graph so built is called the *(undirected) dependency graph* of the system (see [9]).

If we denominate this graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$ is the vertex set and E is the edge set, then, for each vertex/entity i, $1 \leq i \leq n$, it is natural to consider that its state $x_i \in \{0, 1\}$. That is, the entity can be activated or deactivated.

On the other hand, for every vertex/entity i and every subset $W \subset V$, we will consider all the vertices that interfere with them. Thus, we denote by

$$A_G(i) = \{j \in V : \{j, i\} \in E\} \tag{1}$$

the set of vertices that are adjacent to the vertex i.

Nevertheless, in many occasions, the process of information exchange is not bidirectional [67]. This situation can be represented by an arc whose initial vertex is the influencing entity and the final vertex corresponds to the influenced entity, obtaining a directed graph or digraph of relations. The directed graph so built will be called the *directed dependency*

graph of the system. In order to unify the notation, it will be also denoted by $G = (V, E)$, although in this case E is a set of arcs instead of edges. With the same aim, given a directed dependency graph and $i \in V$, $A_G(i)$ will stand for the set of vertices $j \in V$ such that there exists an arc from j to i.

Recalling that, given vertices $i, j \in V$, the *distance* $d(i, j)$ between them is defined as the length of the shortest path from i to j, and the *diameter* of the digraph as

$$\operatorname{diam}(G) = \max\{d(i, j) : i, j \in V\}. \quad (2)$$

2.1.2. Boolean Algebra and Boolean Functions. As the natural way to consider if the state of any entity is activated or deactivated, the basic Boolean algebra $\{0, 1\}$ will play a fundamental role for the theoretical foundations of the model. Moreover, the (general) structure of Boolean algebra is essential to naturally capture the situation where entities are composed by some subentities, such that each of them can be activated or deactivated.

Mathematically, a *Boolean algebra* can be defined [2] as an algebraic structure $(B, \wedge, \vee, ', O, I)$ where B is a set, $\wedge, \vee$ are two inner operations defined on B, $'$ is an application from B to B, and O, I are two distinguished elements of B such that they satisfy certain laws (idempotence, commutative, associative, absorption, distributive, existence of neutral and universal elements, and existence of complement element).

It is common to denote the operation $\vee$ by $+$ and to call it *addition operation* of the Boolean algebra. Likewise, the operation $\wedge$ is usually denoted by $\cdot$ and called *multiplication operation* of the Boolean algebra.

It is natural to introduce the algebraic structure of Boolean algebra as a *Boolean Lattice* which is bounded, distributive, and complemented [1]. In fact, in [2] a definition where only commutative, idempotence, distributive, and complement laws are required is given, which fits better the original definition by George Boole.

If B is any Boolean algebra, then $B^n = \{(x_1, \ldots, x_n) : x_i \in B, \ \forall i\}$ is also a Boolean algebra, where the operations are performed coordinate by coordinate; that is,

(i) $(x_1, \ldots, x_n) \vee (y_1, \ldots, y_n) = (x_1 \vee y_1, \ldots, x_n \vee y_n)$,

(ii) $(x_1, \ldots, x_n) \wedge (y_1, \ldots, y_n) = (x_1 \wedge y_1, \ldots, x_n \wedge y_n)$,

(iii) $(x_1, \ldots, x_n)' = (x_1', \ldots, x_n')$.

In this Boolean algebra, the neutral and universal elements are given, respectively, by $\mathbf{O} = (O, \ldots, O)$ and $\mathbf{I} = (I, \ldots, I)$.

Two Boolean algebras $(B_1, \wedge, \vee, ', O, I)$ and $(B_2, \curlywedge, \curlyvee, `, \widetilde{O}, \widetilde{I})$ are *isomorphic* (as Boolean algebras) [2] if there is a bijection $\varphi : B_1 \to B_2$ such that, for all $x, y \in B_1$, it satisfies

(i) $\varphi(x \vee y) = \varphi(x) \curlyvee \varphi(y)$,

(ii) $\varphi(x \wedge y) = \varphi(x) \curlywedge \varphi(y)$,

(iii) $\varphi(x') = \varphi(x)`$.

In particular, we have that $\varphi(O) = \widetilde{O}$ and $\varphi(I) = \widetilde{I}$.

One of the most important facts in the algebraic structure of Boolean algebra is the existence of some basic elements, named *atoms* of the Boolean algebra, which allow us to *control* the rest of the elements of the structured and focus on them the essence of the structure. An atom is every element $a \in B$, different from the neutral one O, which verifies that $x \wedge a = x$ if and only if $x = a$ or $x = O$.

The atoms allow us to express any element of the Boolean algebra, different from the neutral one O, as a disjunction of them in a unique way (up to the order). Moreover, the well-known Stone's theorem (see [2]) asserts that all the Boolean algebras with the same number of atoms are isomorphic. In particular, every Boolean algebra with p atoms is isomorphic to the Cartesian product of p copies of the basic algebra $\{0, 1\}$, what is essential for obtaining more general results on this model [65]. As a corollary, if a Boolean algebra B has p atoms, then B has 2^p elements.

Another important feature is the *duality principle* in Boolean algebras, which means that if a statement is the consequence of the definition of Boolean algebra, then the dual statement is also true. The dual of a statement in a Boolean algebra is the statement obtained by interchanging the operations $\vee$ and $\wedge$ and the elements O and I in the original statement [2].

Concerning Boolean algebras, another important concept throughout this work is the following.

Definition 1. Let B be a Boolean algebra and $n \in \mathbb{N}$. A *Boolean function* of n variables is a function of the form

$$L : B^n \longrightarrow B, \quad (3)$$

where $L(x_1, x_2, \ldots, x_n) \in B$ is obtained from $x_1, x_2, \ldots, x_n \in B$ using the inner operations $\wedge, \vee$ defined on B, the application $'$ from B to B, and the elements $O, I \in B$.

A Boolean function describes how to determine a Boolean output from some Boolean inputs. Thus, such functions play a fundamental role in questions as design of circuits or computer processes [68]. In our context, they correspond to components of the *evolution operator* of the dynamical system.

An important point is that the set of all the Boolean functions of n variables is also a Boolean algebra (see [2]).

Maxterms and minterms are both special cases of Boolean functions.

Definition 2. A Boolean function of n variables, $x_1, x_2, \ldots, x_n$, that only uses the disjunction operator $\vee$, where each of the n variables appears once in either its direct or its complemented form, is called a *maxterm*.

In this sense, the simplest maxterm corresponds to the one where each of the n variables appears once in its direct form; that is,

$$\operatorname{OR}(x_1, x_2, \ldots, x_n) = x_1 \vee x_2 \vee \cdots \vee x_n. \quad (4)$$

With all the variables in their complemented form, we have the maxterm NAND:

$$\operatorname{NAND}(x_1, x_2, \ldots, x_n) = x_1' \vee x_2' \vee \cdots \vee x_n'. \quad (5)$$

Minterm is the dual concept of maxterm, changing the disjunction operator for the conjunction one.

Definition 3. A Boolean function of n variables that only uses the conjunction operator $\wedge$, where each of the n variables appears once in either its direct or its complemented form, is called a *minterm*.

The simplest minterm corresponds to the one where each of the n variables appears once in its direct form; that is,

$$\mathrm{AND}\left(x_1, x_2, \ldots, x_n\right) = x_1 \wedge x_2 \wedge \cdots \wedge x_n. \tag{6}$$

With all the variables in their complemented form, we have the minterm NOR:

$$\mathrm{NOR}\left(x_1, x_2, \ldots, x_n\right) = x_1' \wedge x_2' \wedge \cdots \wedge x_n'. \tag{7}$$

As one can check, there exist exactly 2^n maxterms of n variables, since a variable in a maxterm expression can be either in its direct or in its complemented form, and dually 2^n minterms. Since the minterms are the atoms of the Boolean algebra which is constituted by all the Boolean functions of n variables, there are exactly 2^{2^n} Boolean functions of n variables.

In particular, see [2, 69], any Boolean function, except $F \equiv I$ (resp., $F \equiv O$), can be expressed in a canonical form as a conjunction (resp., disjunction) of maxterms (resp., minterms). Therefore, it is natural to begin the study of the dynamics with these basic Boolean functions.

2.1.3. Dynamical Systems. Mathematically, a *dynamical system* is defined as a triple (X, T, F) where X is the state space, T is the time set, and F is the evolution operator, verifying the following.

(1) X is a set.

(2) T is a number set.

(3) $F : T \times X \to X$ is a map satisfying

$$\begin{gathered} F(0, x) = x \quad \forall x \in X \\ F(t, F(s, x)) = F(t + s, x) \quad \forall t, s \in T,\ \forall x \in X. \end{gathered} \tag{8}$$

The set X is often a metric space in order to determine the distances between two different states, and it is also called *phase space*, due to classical mechanics.

The most general way in which the evolution operator is given is by means of a family of maps depending on t:

$$F^t : X \longrightarrow X \tag{9}$$

which transforms any initial state x^0 into some state x^t at time t; that is $F^t(x^0) = x^t$.

Depending on the time set T, it is distinguished between discrete and continuous dynamical systems:

$$\begin{gathered} T = \mathbb{Z} \quad \text{or} \quad \mathbb{N} \cup \{0\} \quad \text{or} \quad \mathbb{Z}^- \cup \{0\} \rightsquigarrow \text{Discrete System} \\ T = \mathbb{R} \quad \text{or} \quad \mathbb{R}^+ \cup \{0\} \quad \text{or} \\ \mathbb{R}^- \cup \{0\} \rightsquigarrow \text{Continuous System.} \end{gathered} \tag{10}$$

In this review, we deal with a special kind of discrete dynamical systems. A discrete dynamical system is fully specified by defining only one map F, named the *time-one map* of the system, since for any other t,

$$F^t = F \circ \overset{t)}{\cdots} \circ F. \tag{11}$$

In our particular case, as the state space is finite, the time-one map can be represented by a table which is called *lookup table* for the state updating.

When the evolution operator is defined for both negative and positive values of the time, the system is called *invertible*. In such a system the initial state defines not only the future states, but its past behavior as well. In our context, the invertibility of the system appears associated to the *reversibility problem* [70, 71].

We actually determine a dynamical system over a (directed or undirected) dependency graph $G = (V, E)$ by associating to each vertex $i \in V$ a state $x_i \in \{0, 1\}$ and a local map f_i defined on the states of the vertices in $A_G(i) \cup \{i\}$ and which returns its new state $y_i \in \{0, 1\}$.

In fact, the evolution or update of a system over a (directed or undirected) dependency graph is implemented by local (Boolean) functions. These local functions are either restrictions of a global function or jointly constitute the (global) evolution operator of the system.

If the states of the entities are updated in a parallel manner, the system is called a *parallel dynamical system* (PDS) [9, 41, 42], while if they are updated in a sequential order, the system is named a *sequential dynamical system* (SDS) (see [9, 40, 46]). More precisely.

Definition 4. Let $G = (V, E)$ be a (directed or undirected) graph with $V = \{1, 2, \ldots, n\}$. Then a map

$$\begin{gathered} F : \{0, 1\}^n \longrightarrow \{0, 1\}^n, \\ F\left(x_1, x_2, \ldots, x_n\right) = \left(y_1, y_2, \ldots, y_n\right), \end{gathered} \tag{12}$$

where y_i is the updated state of the entity/vertex i by applying locally the function F over the states of the vertices in $\{i\} \cup A_G(i)$, constitutes a discrete dynamical system called a *parallel (discrete) dynamical system (PDS)* over G with evolution operator F, which will be denoted by $[G, F]$ or F-PDS when specifying the dependency graph is not necessary.

Definition 5. Let $G = (V, E)$ be a (directed or undirected) graph with $V = \{1, 2, \ldots, n\}$ and $\pi = \pi_1 \pi_2 \cdots \pi_n$ a permutation on V. Then a map

$$[F, \pi] = F_{\pi_n} \circ \cdots \circ F_{\pi_2} \circ F_{\pi_1} : \{0, 1\}^n \longrightarrow \{0, 1\}^n, \tag{13}$$

where $F_{\pi_i} : \{0, 1\}^n \to \{0, 1\}^n$ is the update function on the state vector $(x_1, x_2, \ldots, x_n)$ which updates the state of the vertex π_i while keeping the other states unchanged, constitutes a discrete dynamical system called a *sequential (discrete) dynamical system (SDS)* over G with evolution operator $[F, \pi]$, which will be denoted by $[G, F, \pi]$ or $[F, \pi]$-SDS when specifying the dependency graph is not necessary.

Basic objects associated to a dynamical system are its orbits and consequently the *phase portrait*, that is, the partitioning of the state space into its orbits.

Definition 6. The *orbit* of a dynamical system (X, T, F) starting at $x^0 \in X$ is the ordered subset of the state space X given by

$$\mathrm{Orb}\left(x^0\right) = \left\{x \in X : x = F^t\left(x^0\right),\ \forall t \in T\right\} \subset X. \quad (14)$$

Orbits of a discrete dynamical system are ordered sequences of states in the phase space that can be enumerated by increasing integers.

It is worthwhile to recall that, given a discrete dynamical system with global evolution function F and an initial state x^0, the following terminology is used.

(i) $F(x^0)$ is called the *successor* of the state x^0. Observe that, since a dynamical system is deterministic, there exists exactly one successor for each initial state, but this initial state could be the successor of more than one state of the system that are called its *predecessors*. When this occurs the system is often called a *dissipative system* and the *in-degree* of the initial state x^0 is the number of its predecessors (*out-degree* is 1 for every initial state).

(ii) If $F(x^0) = x^0$, then x^0 is called a *fixed point* of F or *equilibrium* state of the system. The notation $FIX[F]$ represents the set of all the fixed points of the dynamical system with global evolution function F.

(iii) If there exists an integer $p > 1$, such that $F^p(x^0) = x^0$ and for any integer $0 < l < p$, $F^l(x^0) \neq x^0$, then x^0 is called a *periodic point* of F or a *periodic state* of the system and p is called the period of the orbit of x^0. The notation $PER[F]$ is adopted to denote the set of all the periodic points of F. Usually, a periodic orbit is called a *cycle*, but we do not use this name in order to avoid confusions with the cycles of the dependency graph.

(iv) When an initial state $x^0 \notin FIX[F] \cup PER[F]$, then x^0 is called a *transient point* of F or a *transient state* of the system. In the case of a finite space state, these points are also called *eventually fixed points* (resp., *eventually periodic points*) if their orbits finally arrive in a fixed point (resp., periodic orbit).

(v) If there does not exist any state x^0, such that $F(x^0) = y$, then y is said to be a *Garden-of-Eden* (GOE) of F. That is, y is a state without predecessors. The set of GOEs of F is denoted by $\mathrm{GOE}[F]$.

According to [6, 72], one of the main goals in the study of a dynamical system is to give a complete characterization of its orbit structure. In this particular case, as the state space of the system is finite, every orbit is periodic or eventually periodic. This means that the evolution from any initial state always reaches an attractor [73]. If the attractor consists of one state, it is called an *attractor point*, whereas if it consists of two or more states, it is said to be an *attractor cycle*. All the states that flow towards an attractor constitute what is called its *basin of attraction*. The time that an initial state takes to reach an attractor is named its *transient*.

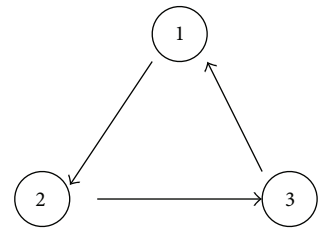

Figure 1: Graphic representation of the dependency digraph G with vertex set $V = \{1, 2, 3\}$ and arc set $E = \{(1, 2), (2, 3), (3, 1)\}$.

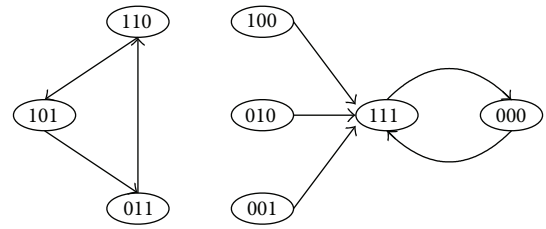

Figure 2: Transition diagram of the PDS $[G, F]$.

Table 1: Lookup table for the time-one-map of the PDS $[G, F]$.

$x_1x_2x_3$	111	110	101	100	011	010	001	000
$y_1y_2y_3$	000	101	011	111	110	111	111	111

The representation of the orbit structure of a system is called *phase portrait*, *phase diagram,* or *transition diagram* of the system. The phase portrait, when possible, gives a complete visual idea of the asymptotic behavior of the system from any initial state.

Example 7. In the next simple example, the different notions related to PDS that were introduced before will be studied in order to clarify them. Let us consider the PDS $[G, F]$ with

$$F : \{0,1\}^3 \longrightarrow \{0,1\}^3, \qquad F\left(x_1, x_2, x_3\right) = \left(y_1, y_2, y_3\right), \quad (15)$$

over the directed graph G shown in Figure 1, where the time-one-map of the system is given by the following lookup table (see Table 1).

This lookup table corresponds to the application of the maxterm NAND as evolution operator, and the transition diagram or phase portrait of the system is the one in Figure 2.

Note that the system is not invertible, since there are 3 initial states that have not any predecessor: 001, 010, and 100. They constitute the GOE of the system and all of them are in the basin of attraction of the attractor cycle $\{111, 000\}$. In fact, they are predecessors of the state 111. Thus, these three states are eventually 2-periodic and the transient to arrive in the attractor cycle is equal to one.

On the other hand, the system does not present any fixed point, although it has a 3-periodic orbit $\{110, 101, 011\}$. As expected, the *Sharkovsky order* [4] for the periodic orbit

structure of a discrete dynamical system where the evolution operator is a continuous function of the interval does not work here.

The absence of fixed points and the presence of periodic orbits of period greater than two in this kind of PDS over directed dependency graphs are two important breakpoints with respect to the pattern followed by those defined over undirected graph (see [41]), as explained in [42].

As shown before, in spite of their relation with classical dynamical systems, the theory and analysis of PDS (and SDS) are based on techniques from algebra, combinatorics, and discrete mathematics in general. In fact, due to the especial *discrete per excellence* character of PDS, some dynamical topics, used to measure the intrinsic stability of a dynamical system, as *sensitivity dependence on initial conditions* or *chaos* [4, 72], are difficult to be translated into this context, since they are established according to the topological properties of the corresponding metric state space.

2.2. Comparative Study with Related Topics. In the specific literature, several topics which are related to PDS appear. In this section, an overview about them is given. The list of related topics presented is reduced to the most similar ones. The purpose is only to provide an introduction in order to compare them with PDS. Specifically, the nearest concepts, namely, *cellular automata* (CA) [21–25] and *Boolean networks* (BN) [14, 28], are reviewed. Other topics related to PDS are, for instance, finite-state machines [74, 75] and Petri nets [76–78], but they will not be discussed here.

2.2.1. Cellular Automata. Following Wolfram [21], a CA consists of a regular uniform lattice (or array), usually infinite in extent, with a discrete variable at each site called *cell*. The state of a CA at a time t is completely determined by the state value x_i of every cell i at such time t. The evolution of this system consists in the discrete time updating of the state value of its cells in a synchronous manner. The updating of the state value of each cell depends on a local function defined on the state values, at the previous time step, of other cells in its *neighborhood*. The neighborhood of a site in the uniform lattice is typically constituted by the site itself and all the immediately adjacent sites. These immediately adjacent cells are determined by the lattice structure and a homogeneous rule for the selection of neighbors.

Thus, CA can be contemplated as a particular case of (infinite) PDS (the concept of PDS can be easily extended to involve an infinite number of entities), if one considers cells as entities. Nevertheless, CA have a more restricted and uniform structure and, in contrast to PDS, CA are usually considered over inf of the system are fixed mple, $\mathbb{Z}^d$, where $d \geq 1$ is said to be the *dimension of the CA*.

The more restricted and uniform structure of CA is reflected by both the neighborhood $A(i)$ of any cell i and the local functions f_i acting on it, which are determined homogeneously. Certainly, in classical CA, every cell i has a neighborhood $A(i)$ which is some sequence of lattice sites, which displays a homogeneous structure. If the CA is defined over $\mathbb{Z}^d$, this homogeneous structure in the neighborhood of any cell i is obtained by means of a (finite) subset $\{j_1, \ldots, j_k\} \subset \mathbb{Z}^d$ which provides a rule to have

$$A(i) = \{i + j_1, i + j_2, \ldots, i + j_k\}. \tag{16}$$

Observe that in a CA over an infinite structure like $\mathbb{Z}^d$, any state of the system is given by a *sequence* $(x_i)_{i \in \mathbb{Z}^d}$, usually named a *configuration*. At this point, one can state the definition of a *d-dimensional cellular automata*, taking into account the (finite) set B of the state values that any cell can have and the rule to define the neighborhood $A(i)$ of any cell i over an infinite lattice $\mathbb{Z}^d$, where the same local function is applied. When the set B is equal to the basic Boolean algebra $B = \{0, 1\}$ the CA is said to be a *Boolean cellular automata*.

One-dimensional (Boolean) CA with two possible values for the state of the cells, 0 and 1, in which the neighborhood of a given site is simply the site itself and the sites immediately adjacent to it on its left and right, are studied in [21]. In such a case, the subset to get a homogeneous neighborhood is $\{-1, 0, 1\} \subset \mathbb{Z}$, and it is called a *three-neighborhood* structure (see Figure 3).

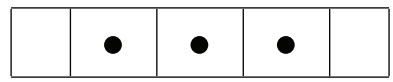

Figure 3: Three-neighborhood structure for one-dimensional CA.

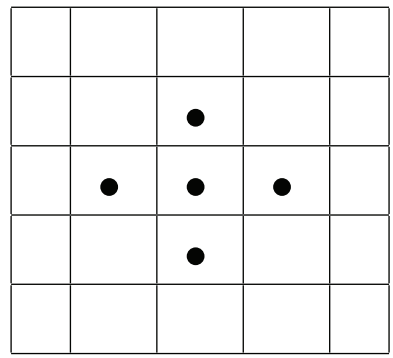

Figure 4: Von Neumann neighborhood structure for two-dimensional CA.

For two-dimensional CA, two of the most used neighborhood structures are the *von Neumann neighborhood* [30] and the *Moore neighborhood* [79] (see Figures 4 and 5). The finite subset to obtain the von Neumann neighborhood is

$$\{(0,0), (-1,0), (0,-1), (1,0), (0,1)\} \subset \mathbb{Z}^2, \tag{17}$$

while the one to get the Moore one is

$$\{(0,0), (-1,0), (0,-1), (1,0), (0,1), (1,1), (-1,-1), (1,-1), (-1,1)\} \subset \mathbb{Z}^2. \tag{18}$$

CA over finite lattices with n sites can also be defined. Two traditional ways to do that are as follows:

(i) by imposing *periodic boundary conditions*, which consists in identifying the cells separated by n positions,

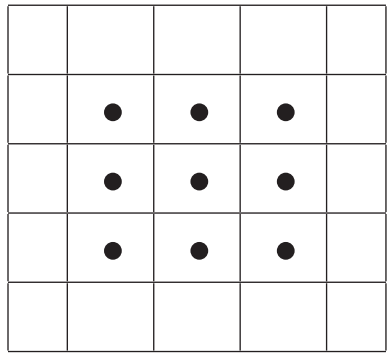

Figure 5: Moore neighborhood structure for two-dimensional CA.

giving a CA over $\mathbb{Z}/n\mathbb{Z}$ [80], an idea which can be extended to higher dimensions, giving a CA over a k-dimensional tori,

(ii) by imposing *zero boundary conditions*, which consists in using a line graph as a lattice and adding two additional extreme vertices whose states are always equal to zero.

Local rules for the updating of Boolean CA are in fact Boolean functions of k variables, where k is the number of neighbors in any neighborhood $A(i)$. Most of the research on CA has been concerned with one-dimensional cases, which are commonly known as *elementary cellular automata* (see [10, 12, 21–23, 30, 49, 71, 79, 81–83]).

For a one-dimensional, three-neighborhood, Boolean CA, a local function has to determine the (output) state value of the central cell, from the (input) state values of the central cell and its two immediately adjacent ones. As we are considering Boolean values, there are only 2^3 possible input state values for any neighborhood in the lattice, namely,

$$111 \quad 110 \quad 101 \quad 100 \quad 011 \quad 010 \quad 001 \quad 000. \tag{19}$$

As from any input state values the output state value can be 0 or 1; there are $2^{2^3} = 256$ distinct local rules for elementary three-neighborhood CA. Traditionally, each of these local rules has a decimal number belonging to $\{0, 1, 2, \ldots, 255\}$ assigned and it is known as the *Wolfram enumeration* of *elementary CA rules*. The procedure to enumerate such rules consists in considering the (output) value states of the triples above in that order and then translating the corresponding binary number into a decimal one. That is, if one has the output value states of the triples above as in Table 2, then the number of the local rule is

$$\sum_{i=0}^{7} \alpha_i \cdot 2^i. \tag{20}$$

This enumeration procedure can be generalized to other classes of neighborhoods in CA.

Thus, the elementary CA rule 0 erases any initial configuration, while 204 does not change any initial configuration; that is, it corresponds to the identity transformation. One of the most important elementary three-neighborhood CA is the one given by the local function *XOR*, that is, the elementary CA rule number 150 (see Table 3).

Table 2: Denomination of the updating of the state of any entity i to establish the procedure to enumerate elementary CA rules.

$x_{i-1}x_ix_{i+1}$	111	110	101	100	011	010	001	000
y_i	α_7	α_6	α_5	α_4	α_3	α_2	α_1	α_0

Table 3: Updating of the state of any entity i corresponding to the elementary CA rule number 150.

$x_{i-1}x_ix_{i+1}$	111	110	101	100	011	010	001	000
y_i	1	0	0	1	0	1	1	0

Its importance is due to its applications to cryptographic protocols (see [10] or [12]).

CA, when finite, can be considered as a special kind of PDS by considering cells as entities. It is easy to understand that the lattice of a CA allows us to infer a PDS dependency graph by considering the cells as entities, being any entity adjacent to those that are its neighbors in the lattice. Even, when infinite, CA can be considered PDS, since the concept of PDS can be easily extended for an infinite number of entities.

Also, CA are updated in a parallel or synchronous manner by applying local functions on a subset that contains the (state value of the) cell. Nevertheless, in the last few years some extensions of the concept of CA considering sequential or asynchronous updating have appeared in the literature (see [84–86]). In fact, the concept of sequential dynamical system [9, 33–35, 40, 44, 45] constitutes a generalization of such CA extension.

However, CA are restricted cases of PDS in several ways. First of all, for a CA seen as a PDS, the dependency graph, which is derived from the lattice and the neighborhood structure, is regular, whereas the graph of a general PDS can be arbitrary. Secondly, CA have the same local function or rule associated with every cell, while general PDS can have distinct local functions to update different entities, which can be the restriction of a global one (see [9, 41]) or independently defined (see [43]). Thus, general PDS can have more involved update schemes.

Although theoretically the set B of state values of any cell can be whichever finite set, most of works in the literature focus on Boolean CA [10, 12, 21–23, 30, 49, 71, 79, 81–83], where $B = \{0, 1\}$. In [87] and more recently in [88], a first approach in the study of CA where B is any finite set is given. On the other hand, in [65], we study completely the problem when the set B is any Boolean algebra in the more general context of PDS.

Summing up, PDS are a generalization of CA, where the state value of each entity is not affected necessarily by its neighbors (see Figure 6), but potentially by any entity in the network, and the updating function f_i associated with an entity i can be given by any arbitrary local function.

2.2.2. Boolean Networks. A Boolean network (BN) of size n and connectivity k consists of n interconnected vertices, each one having k inputs. The update of the state of any

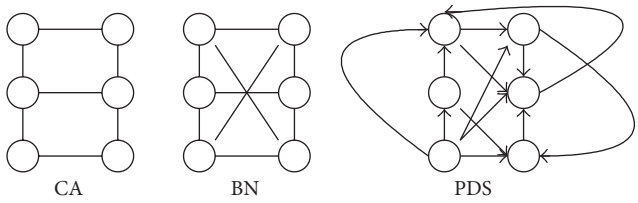

Figure 6: Graphic representation of comparative dependency graphs for CA, BN, and PDS.

vertex is determined by the (directed) dependency relations and local rules which are given by Boolean functions. Hence, BN are a generalization of (finite) Boolean CA but, at the same time, a particular case of PDS by considering nodes as entities. One of the main differences with CA is that, in BN, the state of each node is not affected necessarily by its neighbors but potentially by any node in the network. Thus, the uniform structure of neighborhood in CA disappears. Nevertheless, some homogeneity remains, since each node is affected by k connections with other (or the same) entities. This homogeneity makes BN a particular case of PDS where the connections can be totally arbitrary (see Figure 6).

Another important difference with CA is that local Boolean functions of k-variables are generated randomly, which provides different lookup tables for each entity for the updating process. This idea has been carried out and extended for PDS in this work in two directions. Firstly, as can be seen in [41], local Boolean functions acting on each entity can have different number of variables (which cannot occur for BN); and secondly, they can be totally independent for each entity.

BN with n nodes and k connectivity are known as *n-k models* or *Kauffman models*, since they were originally developed by him in [26] when modeling regulatory networks of (on-off) genes. For this reason, nodes of BN are often called *genes*. In particular, Kauffman models on complete networks are usually called *random maps*.

As in PDS, the state space is finite with 2^n possible states. Therefore, whichever the initial state is, eventually a state will be repeated. However, the number of *n-k* models, when n and k are fixed, is lower than the number of possible dependency graphs that determine a PDS.

As for the case of CA, originally the updating of BN was synchronous, but, also in this case, several recent works provide some extensions of the concept, considering asynchronous updating schedule [11, 85, 86]. Also in this case, the concept of sequential dynamical system constitutes a generalization of such *asynchronous Boolean networks*.

Although first Kauffman models were deterministic, in the last decade some authors have studied an extension of the concept considering a probabilistic updating schedule [14, 66, 89]. In this context, they are named *probabilistic Boolean networks* and cannot be described from the perspective of dynamical systems, with *Markov chains* being the natural framework to set out their evolution. This idea of extension was provided for the first time in [66] and it will be explained in the following paragraphs.

Let $0 \leq p \leq 1$, i a gene or node of a BN, and denote by f_i a local function to update the state value of i for all $i = 1, \ldots, n$. On the other hand, let f_i' be another local function related to the updating of the node i. If any node i is updating by means of f_i with probability p and by means of f_i' with probability $(1 - p)$, then a basic probabilistic BN is obtained. Thus, this stochastic construction may be viewed as a weighted superposition of two deterministic BN, that is, the one obtained with the global evolution operator $F = (f_1, f_2, \ldots, f_n)$ and the other one achieved with the time-one-map $F' = (f_1', f_2', \ldots, f_n')$. That is, by updating the BN with the evolution operator F, a transition diagram could be determined, namely Γ, while updating with F' another transition diagram, namely Γ', could be also obtained. Then, by superposing the transition diagrams (which are in fact direct graphs), one can construct a digraph, assigning the weight p to the arcs which are only in Γ, the weight $1 - p$ to the arcs which are only in Γ', and the weight 1 to the arcs which are in both diagrams. This fused weighted digraph is often called the *probabilistic phase space* and its weighted matrix can be considered as the corresponding Markov chain matrix. The (probabilistic) evolution of the BN from an initial state can be then identified with a random walk over this probabilistic phase space that starts at this initial state of the BN.

The construction explained before is basic in the sense that only two kinds of functions with two probabilities are considered to update the nodes of the BN. Obviously, more functions could be considered and then the probabilistic phase space would be the result of the superposition of all the transitions diagrams obtained deterministically with any global operator, considering for any arc that appears in more than one of this transitions diagrams the weight resulting of the sum of all the probabilities assigned to any of the diagrams where this arc appears.

Finally, observe that the PDS notion could be extended in order to capture probabilistic updating in the same sense that probabilistic BN do. Nevertheless, in this case, they would lose the structure of (deterministic) dynamical systems.

3. Analytical Results on the Orbital Structure of PDS

3.1. PDS over Undirected Dependency Graphs. This section is devoted to review the orbit structure of parallel discrete dynamical systems with maxterms and minterms Boolean functions as global evolution operators. As a result, it is shown that the orbit structure does not remain when the system is perturbed.

In [9], it is proved that, for a parallel dynamical system OR-PDS associated with the maxterm OR, all the orbits of the system are fixed points or eventually fixed points. More precisely, the system has exactly two fixed points and the maximum number of iterations needed by an eventually fixed point to reach the corresponding fixed one is at most as large as the diameter of the dependency graph.

Dually, for a parallel dynamical system AND-PDS associated with the minterm AND, all the orbits of the system are fixed points or eventually fixed points, and the system has exactly two fixed points and the maximum number of

iterations needed by an eventually fixed point to reach the corresponding fixed one is at most as large as the diameter of the dependency graph.

Observe that the orbit structures of OR-PDS and AND-PDS do not depend on the dependency graphs over which they are defined, covering all the possible casuistry of them.

In [41], it is proved that, for a parallel dynamical system MAX-PDS associated with an arbitrary maxterm MAX, all the periodic orbits of this system are fixed points or 2-periodic orbits, while the rest of the orbits are eventually fixed points or eventually 2-periodic orbits. In fact, from the demonstration, the following can be deduced.

(i) If all the variables are in their direct form, only fixed points or eventually fixed points can appear.

(ii) If all the variables are in their complemented form, only 2-periodic orbits or eventually 2-periodic orbits can appear.

(iii) Otherwise, both kinds of situations coexist.

As before, dually, for a parallel dynamical system MIN-PDS associated with an arbitrary minterm MIN, all the periodic orbits of this system are fixed points or 2-periodic orbits, while the rest of the orbits are eventually fixed points or eventually 2-periodic orbits.

These results provide a complete characterization of the orbit structure of parallel discrete dynamical systems with maxterms and minterms Boolean functions as global evolution operators. In particular, one can infer that when other maxterms (resp., minterms) different from OR (resp., AND) are considered as global functions to define the evolution of a system, the dynamics can change, giving as a result a certain absence of structural stability of the system so perturbed.

3.2. PDS over Directed Dependency Graphs. In this section, the orbit structure of parallel discrete dynamical systems over directed dependency graphs (PDDS) (the abbreviation PDDS will be written for the singular and plural forms of the corresponding term, since it seems better from an aesthetic point of view) with Boolean functions as global evolution operators is reviewed. In this sense, for the cases corresponding to the simplest Boolean functions AND and OR, it is explained that only fixed or eventually fixed points appear, as occurs over undirected dependency graphs. However, for general Boolean functions, it is shown that the pattern found for the undirected case breaks down.

In [42], it is proved that for a parallel dynamical system OR-PDDS over a directed dependency graph associated with the maxterm OR, all the orbits of this system are fixed points or eventually fixed points. Dually, for a parallel dynamical system AND-PDDS over a directed dependency graph associated with the minterm AND, we have also that all the orbits of this system are fixed points or eventually fixed points.

Nevertheless, comparing with the undirected case studied in the previous section, for the directed one the number of fixed points can increase significantly, except if the digraph is strongly connected (i.e., if for all pairs of vertices $i, j \in V$, there exists a path from i to j and another one from j to i), where only two equilibrium states are possible, namely, the state with all the entities activated and the one with all the entities deactivated. In general, this number of possibilities depends on the structure of the directed dependency graph of the system and it is not possible to give a generic result as in the undirected case.

Besides, also in [42], it is demonstrated that for the general case of PDDS associated with maxterms and minterms as global evolution operators any period can appear, breaking the pattern found for the undirected case where only fixed points or 2-periodic orbits can exist [41]. In order to prove that, a method is developed to provide, for every given period, a PDDS which presents a periodic orbit of such a period.

Subsequent research [64] allows us to check that this breakdown is due to the existence of directed cycles in the dependency digraph, while for PDDS over *acyclic* dependency digraphs the periodic orbits continue being only fixed points or 2-periodic ones. Moreover, the orbit structure of special digraph classes as *line digraphs, arborescences*, and *star digraphs* is similar and it is possible to specify the number of iterations needed by an eventually periodic point to reach the corresponding periodic orbit.

However, it is not easy to control the orbit structure when the dependency graph has cycles. In such a case, the orbital structure depends on both the global evolution operator and the structure of the digraph. In particular, in [64], it is shown that NAND-PDDS over circle digraphs can present periodic orbits of any period except fixed points and periods 4 and 6, and the same occurs for NOR-PDDS.

4. Extensions of the Concept of PDS

In this section, two extensions on parallel dynamical systems are reviewed. The first one is related to the manner of defining the evolution update and the second one to consider that the states of the entities can take values in an arbitrary Boolean algebra.

4.1. PDS on Independent Local Functions. When Kauffman introduced BN for the first time [26], he contemplated the possibility of updating any entity using independent local functions. This idea is extended to PDS in [43], since Boolean functions acting on each entity can have different number of variables (which cannot occur for BN) due to the random (connections) structure of the dependency graph. This extension of the update method widely generalizes the traditional one, where only a global Boolean function is considered for establishing the evolution operator of the system. Besides, this analysis allows us to show a richer dynamics in these new kinds of parallel dynamical systems.

In particular, for a PDS $[G, \{f_i\}]$ over an undirected or directed graph G where each local function f_i is either AND or OR, it is demonstrated that all the orbits of this system are fixed points or eventually fixed points. Although the orbit structure coincides with the corresponding one for a PDS where the updating global Boolean function is either AND or OR, for a PDS $[G, \{f_i\}]$ as the one described above, the number of fixed points of the system (and therefore

the complexity of the system) may increase depending on the election of the local functions f_i and the structure of the dependency graph. In order to illustrate this, we show the following example.

Example 8. Consider the undirected dependency graph given by $V = \{1, 2, 3\}$ and $E = \{\{1, 2\}, \{2, 3\}\}$. If we consider the PDS defined by the global Boolean function AND (resp., OR), then the fixed points are $(1, 1, 1)$ and $(0, 0, 0)$. On the other hand, if we consider the PDS defined by the local functions

$$f_1 = \text{AND}, \quad f_2 = \text{OR}, \quad f_3 = \text{AND} \tag{21}$$

then the fixed points are $(1, 1, 1)$, $(0, 0, 0)$, $(0, 1, 0)$, $(1, 1, 0)$, and $(0, 1, 1)$.

Moreover, for a PDS $[G, \{f_i\}]$ over an undirected graph where each local function f_i is either NOR or NAND, the periodic orbits are fixed points or 2-periodic orbits and the only possible fixed point is the one where $x_i = 0$ if $f_i = \text{NOR}$ and $x_i = 1$ if $f_i = \text{NAND}$. Nevertheless, a PDS over a directed graph where each local function f_i is either NOR or NAND can present periodic orbits of any period $n \in \mathbb{N}$.

Finally, also in [43], it is demonstrated that for a PDS $[G, \{f_i\}]$ over an undirected graph where each local function $f_i \in \{\text{AND}, \text{OR}, \text{NOR}, \text{NAND}\}$, the periodic orbits of this system are fixed points or 2-periodic orbits, while over a directed graph periodic orbits of any period $n \in \mathbb{N}$ can appear.

4.2. PDS with General Boolean State Values. In [65], another extension of PDS is provided, by considering that the states of the vertices can take values in an arbitrary Boolean algebra B of 2^p elements, $p \in \mathbb{N}$, $p \geq 1$. This situation naturally appears, for instance, when each entity is composed of p subentities which can be activated or deactivated, and consequently the states of the entities belong to the Boolean algebra $\{0, 1\}^p$. This definition widely extends the traditional one where it is assumed that every entity can take values in the simplest Boolean algebra $\{0, 1\}$. To study the orbit structure of these more general PDS, in view of the Stone's Theorem (see [2]), an isomorphism as Boolean algebras of B and $\{0, 1\}^p$ is considered. Then, since the operations and complement elements are obtained coordinate by coordinate in $\{0, 1\}^p$, it is shown how one can translate all the results previously achieved to this new context.

5. Algorithms for the Computation of Orbits

As said before, in [42], it is demonstrated that for the general case of PDDS associated with maxterms and minterms as global evolution operators any period can appear, breaking the pattern found for the undirected case. Since we have a finite state space, it is obvious that every orbit is either periodic or eventually periodic. However, it is not so easy to determine a priori the different coexistent periods of its orbits, because it depends fundamentally on the structure of the directed dependency graph. In this sense, in [90], algorithms for computing the (eventually) period of any orbit are given.

Since the coexistence of periodic orbits depends on the number of entities, their connections, and the Boolean operator, several matrix algorithms for the computation of orbits in PDDS over directed dependency graph are developed in [90]. These algorithms constitute a new tool for the study of orbits of these dynamical systems. The algorithms are also valid for PDS over undirected dependency graphs. Likewise, the methods are extended to the case of SDS, even when the local functions that are considered for the updating of any entity are independent.

These matrix methods of computation of orbits provide an especial mathematical linearization of the system, by means of particular products of the adapted adjacency matrix of the dependency graph and the state vectors.

We think the algorithms established can help to infer some analytical results on the orbit structure of PDS and SDS over special classes of digraphs, since they can be used to reveal some orbit patterns.

6. Conclusions and Future Research Directions

The results on PDS reviewed provide a full characterization of the orbit structure of PDS over undirected graphs covering all the casuistry, since it studies the behavior of any parallel dynamical system regardless the type of graph, the number of entities, and the relationships among them. Specifically, when the evolution functions are OR or AND or a combination of both, the orbits are fixed points. But when considering other maxterm or minterms dynamics can change, due to the appearance of 2-periodic orbits, resulting in a certain absence of structural stability when systems are slightly perturbed.

On the other hand, essential properties of the orbit structure of PDS over directed dependency graphs are achieved. In fact, evolutions with OR and AND continue giving only fixed or eventually fixed points. But, on the contrary, for general maxterm or minterm functions, it is shown that any period can appear, so breaking the pattern found for the undirected case. Actually, several PDS over especial digraph class are characterized. Besides, it is shown that this disappearance of the general pattern is due to presence of cycles in the dependency digraph.

Motivated by these findings, computation methods for calculating orbits in PDS are also developed. Besides, two extensions of the PDS concept are shown.

The study developed so far opens several future research directions. Most of the problems that can arise after this study of PDS correspond to extensions of the model, some of which have already begun to be studied in some works in the narrower context of CA or BN. It would be interesting to extend this study to the more general context of PDS (or SDS).

One of the initial questions posed by this work is to check if the results for PDS are transferable to the case of SDS. Paper [9] shows a first approach to the problem, since the authors carried out a parallel study of the orbital structure of PDS and SDS with evolution OR and NOR operators, obtaining similar results in both cases, that is, fixed points when the evolution operator is OR and periodic points of period 2 when the evolution operator is NOR. By duality, these results are also valid for the case of AND and NAND operators.

Results in [41] solve the problem of the orbital structure of PDS over undirected graphs for which the evolution operator is any maxterm or minterm, including the former OR, AND, NOR, and NAND. Therefore, a possible new line of research in view of this work is to study if the results obtained for PDS can be translated to the case of SDS, fundamentally in the case in which the systems are defined over undirected dependency graphs.

In the case of directed dependency graphs, the extension of the results to SDS seems much more complicated, as one can check with simple examples.

When modeling discrete processes, it can occur that the entities are updated in a mixed manner between parallel and sequential one. This happens when there exists a partition of the set of entities V of the system such that every set of this partition consists of entities that are updated synchronously, but the updating of these sets is asynchronous. That is, once the partition of the vertex set is known, the subsets can be ordered by a permutation that indicates how these subsets are sequentially updated. This means that given an ordered partition $V_1, V_2, \ldots, V_k \subset V$, firstly the entities of the subset V_1 are updating in a parallel manner; then the entities of V_2 are also updated in a parallel manner, but taking into account the new states values of the entities in V_1, and so on. These systems can be called *mixed dynamical systems* (MDS). This new updating scheme could be used as an intermediate step between PDS and SDS. In fact, SDS can be considered as MDS where the partition of the set V is given by all the subsets with only one element. This notion should be investigated not only for the natural interest to model mixed computer processes, but also for the possible repercussions in the translation of results from PDS to SDS.

The original definition of CA [21] contemplates the possibility that the cells take state values in a finite set, although subsequently the majority of studies have been made in the case of Boolean CA. The studies that have appeared so far in this direction ([87, 88]) consider some uniformity, since every entity can have all the state values in the finite set. This can have the physical sense of assuming that the entities can have different levels of intensity. In this sense, considering that the entities of a system may not have necessarily the same number of states, it could be encouraging to establish a consistent update schedule when the states of the different entities of the system do not belong to the same finite set.

In [65], we study completely the problem when the set B is any Boolean algebra in the more general context of PDS. This constitutes an important issue, since it can model the case where entities are composed of a finite number of parts that can be activated or deactivated. In this context, the problem above concerning entities with different set of state values is maybe more suitable, since any set with structure of Boolean algebra is isomorphic to a subalgebra of $\{0, 1\}^n$ and then one could establish a surjection from this algebra to one smaller in order to make a correspondence among states of this bigger Boolean algebra and the smaller one.

On the other hand, to give a fair explanation about how a system evolves when the entities can have state values in a finite set would open the door to the definition and posterior study of a fuzzy version of PDS.

Of course, one can also think in the possibility of considering an infinite set of state values for the entities. But, again, this is not a natural situation in the processes modeled by PDS.

Also in the original definition of CA in [21], it is allowed to have an infinite number of cells in the system. Although computer processes depend on a finite number of entities, the concept of PDS and the results obtained in this work can be extended to the case of infinite number of entities.

In the same sense as before, one can also think in the possibility of considering an infinite number of entities and a finite or infinite set of state values for the entities.

The notion of dynamically equivalent systems allows us to describe theoretically the class of all dynamically equivalent systems. Therefore, it should be desirable to find out some properties in the dependency graph and the evolution operator which provide or characterize PDS equivalent to a given one. A first approach in this sense appears in [91], where finite sequential dynamical systems on binary strings are studied. Several equivalence relations notions as *isomorphic* and *stably isomorphic* are introduced, and the resulting equivalence classes are studied. In this same sense, researching in the conditions that provoke bifurcations is a compiling need.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work has been partially supported by Grants MTM2011-23221.

References

[1] C. L. Liu, *Elements of Discrete Mathematics*, McGraw-Hill, New York, NY, USA, 1985.

[2] K. Rosen, *Discrete Mathematics and Its Applications*, McGraw-Hill, Boca Raton, Fla, USA, 2011.

[3] D. K. Arrowsmith and C. M. Place, *An Introduction to Dynamical Systems*, Cambridge University Press, Cambridge, UK, 1990.

[4] R. L. Devaney, *An Introduction to Chaotic Dynamical Systems*, Addison-Wesley, Ann Arbor, Mich, USA, 1989.

[5] M. W. Hirsch and S. Smale, *Differential Equations, Dynamical Systems and Linear Algebra*, Academic Press, New York, NY, USA, 1974.

[6] Y. A. Kuznetsov, *Elements of Applied Bifurcation Theory*, Springer, New York, NY, USA, 2004.

[7] G. Boole, *An Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities*, Cambridge University Press, Cambridge, Mass, USA, 1854.

[8] C. E. Shannon, *A symbolic analysis of relay and switching circuits [M.S. thesis]*, MIT Press, Cambridge, Mass, USA, 1940.

[9] C. L. Barrett, W. Y. Chen, and M. J. Zheng, "Discrete dynamical systems on graphs and Boolean functions," *Mathematics and Computers in Simulation*, vol. 66, no. 6, pp. 487–497, 2004.

[10] A. Fúster-Sabater and P. Caballero-Gil, "On the use of cellular automata in symmetric cryptography," *Acta Applicandae Mathematicae*, vol. 93, no. 1–3, pp. 215–236, 2006.

[11] F. Greil and B. Drossel, "Dynamics of critical Kauffman networks under asynchronous stochastic update," *Physical Review Letters*, vol. 95, no. 4, Article ID 048701, 2005.

[12] S. Nandi, B. K. Kar, and P. P. Chaudhuri, "Theory and applications of cellular automata in cryptography," *IEEE Transactions on Computers*, vol. 43, no. 12, pp. 1346–1357, 1994.

[13] Z. Roka, *Cellular automata on cayley graphs [Ph.D. thesis]*, École Normale Supérieure de Lyon, Lyon, France, 1994.

[14] I. Shmulevich, E. R. Dougherty, and W. Zhang, "From boolean to probabilistic boolean networks as models of genetic regulatory networks," *Proceedings of the IEEE*, vol. 90, no. 11, pp. 1778–1792, 2002.

[15] Z. Toroczkai and H. Guclu, "Proximity networks and epidemics," *Physica A*, vol. 378, no. 1, pp. 68–75, 2007.

[16] A. Ilachinski, *Cellular Automata. A Discrete Universe*, World Scientic, Singapore, 2001.

[17] J. Kari, "Theory of cellular automata: a survey," *Theoretical Computer Science*, vol. 334, no. 1–3, pp. 3–33, 2005.

[18] J. Maynard-Smith, *Evolution and the Theory of Games*, Cambridge University Press, Cambridge, Mass, USA, 1982.

[19] P. Sarkar, "A brief history of cellular automata," *ACM Computing Surveys*, vol. 32, no. 1, pp. 80–107, 2000.

[20] J. L. Schiff, *Cellular Automata*, John Wiley & Sons, New York, NY, USA, 2008.

[21] S. Wolfram, "Statistical mechanics of cellular automata," *Reviews of Modern Physics*, vol. 55, no. 3, pp. 601–644, 1983.

[22] S. Wolfram, "Universality and complexity in cellular automata," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1-2, pp. 1–35, 1984.

[23] S. Wolfram, "Algebraic properties of cellular automata," *Physica D*, vol. 10, pp. 1–25, 1984.

[24] S. Wolfram, *Theory and Applications of Cellular Automata*, vol. 1 of *Advanced Series on Complex Systems*, World Scientific Publishing, Singapore, 1986.

[25] S. Wolfram, *Cellular Automata and Complexity*, Addison-Wesley, New York, NY, USA, 1994.

[26] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, vol. 22, no. 3, pp. 437–467, 1969.

[27] S. A. Kauffman, *Origins of Order: Self-Organization and Selection in Evolution*, Oxford University Press, Oxford, UK, 1993.

[28] R. Serra, M. Villani, and L. Agostini, "On the dynamics of random Boolean networks with scale-free outgoing connections," *Physica A: Statistical Mechanics and its Applications*, vol. 339, no. 3-4, pp. 665–673, 2004.

[29] I. Shmulevich and S. A. Kauffman, "Activities and sensitivities in Boolean network models," *Physical Review Letters*, vol. 93, no. 4, 2004.

[30] J. von Neumann, *Theory of Self-Reproducing Automata*, University of Illinois Press, Chicago, Ill, USA, 1966.

[31] M. Gardner, "The fantastic combination of John Conway's new solitaire game 'life'," *Scientific American*, vol. 223, pp. 120–123, 1970.

[32] J. H. Conway, "What is life?" in *Winning Ways for Your Mathematical Plays*, Academic Press, New York, NY, USA, 1982.

[33] C. L. Barrett and C. M. Reidys, "Elements of a theory of computer simulation I: sequential CA over random graphs," *Applied Mathematics and Computation*, vol. 98, no. 2-3, pp. 241–259, 1999.

[34] C. L. Barret, H. S. Mortveit, and C. M. Reidys, "Elements of a theory of computer simulation II," *Applied Mathematics and Computation*, vol. 107, pp. 121–136, 2002.

[35] C. L. Barrett, H. S. Mortveit, and C. M. Reidys, "Elements of a theory of simulation III: equivalence of SDS," *Applied Mathematics and Computation*, vol. 122, no. 3, pp. 325–340, 2001.

[36] C. L. Barrett, H. S. Mortveit, and C. M. Reidys, "ETS IV: sequential dynamical systems: fixed points, invertibility and equivalence," *Applied Mathematics and Computation*, vol. 134, no. 1, pp. 153–171, 2003.

[37] A. Bagrodia, K. M. Chandy, and W. T. Liao, "A unifying framework for distributed simulation," *ACM Transactions on Modeling and Computer Simulation*, vol. 1, no. 4, pp. 348–385, 1991.

[38] D. Jefferson, "Virtual time," *ACM Transactions on Programming Languages and Systems*, vol. 7, no. 3, pp. 404–425, 1985.

[39] R. Kumar and V. Garg, *Modeling and Control of Logical Discrete Event Systems*, Kluwer Academic, Dordrecht, The Netherlands, 1995.

[40] H. S. Mortveit and C. M. Reidys, "Discrete, sequential dynamical systems," *Discrete Mathematics*, vol. 226, no. 1–3, pp. 281–295, 2001.

[41] J. A. Aledo, S. Martínez, F. L. Pelayo, and J. C. Valverde, "Parallel discrete dynamical systems on maxterm and minterm Boolean functions," *Mathematical and Computer Modelling*, vol. 55, no. 3-4, pp. 666–671, 2012.

[42] J. A. Aledo, S. Martínez, and J. C. Valverde, "Parallel dynamical systems over directed dependency graphs," *Applied Mathematics and Computation*, vol. 219, no. 3, pp. 1114–1119, 2012.

[43] J. A. Aledo, S. Martínez, and J. C. Valverde, "Parallel discrete dynamical systems on independent local functions," *Journal of Computational and Applied Mathematics*, vol. 237, no. 1, pp. 335–339, 2013.

[44] R. Laubenbacher and B. Pareigis, "Decomposition and simulation of sequential dynamical systems," *Advances in Applied Mathematics*, vol. 30, no. 4, pp. 655–678, 2003.

[45] R. Laubenbacher and B. Pareigis, "Update schedules of sequential dynamical systems," *Discrete Applied Mathematics*, vol. 154, no. 6, pp. 980–994, 2006.

[46] H. S. Mortveit and C. M. Reidys, *An Introduction to Sequential Dynamical Systems*, Springer, New York, NY, USA, 2007.

[47] I. Shmulevich, S. A. Kauffman, and M. Aldana, "Eukaryotic cells are dynamically ordered or critical but not chaotic," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 38, pp. 13439–13444, 2005.

[48] A. Deutsch and S. Dormann, *Cellular Automaton Modelling of Biological Pattern Formation*, Birkhäuser, Boston, Mass, USA, 2004.

[49] P. Hogeweg, "Cellular automata as a paradigm for ecological modeling," *Applied Mathematics and Computation*, vol. 27, no. 1, pp. 81–100, 1988.

[50] U. Dieckman, R. Law, and J. A. J. Metz, *The Geometry of Ecological Interactions. Simplifying Spatial Complexity*, Cambridge University Press, Cambridge, UK, 2000.

[51] J. Hofbauer and K. Sigmund, *Evolutionary Games and Population Dynamics*, Cambridge University Press, Cambridge, UK, 2003.

[52] G. Chiaselotti, G. Marino, P. A. Oliverio, and D. Petrassi, "A discrete dynamical model of signed partitions," *Journal of Applied Mathematics*, vol. 2013, Article ID 973501, 10 pages, 2013.

[53] C. Bisi and G. Chiaselotti, "A class of lattices and boolean functions related to the Manickam-Miklös-Singhi conjecture," *Advances in Geometry*, vol. 13, no. 1, pp. 1–27, 2013.

[54] C. Bisi, G. Chiaselotti, and P. A. Oliverio, "Sand piles models of signed partitions with d piles," *ISRN Combinatorics*, vol. 2013, Article ID 615703, 7 pages, 2013.

[55] G. Chiaselotti, T. Gentile, G. Marino, and P. A. Oliverio, "Parallel rank of two sandpile models of signed integer partitions," *Journal of Applied Mathematics*, vol. 2013, Article ID 292143, 12 pages, 2013.

[56] G. Chiaselotti, T. Gentile, and P. A. Oliverio, "Parallel and sequential dynamics of two discrete models of signed integer partitions," *Applied Mathematics and Computation*, vol. 232, pp. 1249–1261, 2014.

[57] G. Chiaselotti, W. Keith, and P. A. Oliverio, "Two self-dual lattices of signed integer partitions," *Applied Mathematics & Information Sciences*, vol. 8, no. 6, pp. 3191–3199, 2014.

[58] B. Chopard and M. Droz, *Cellular Automata for Modeling Physics*, Cambridge University Press, Cambridge, UK, 1998.

[59] L. B. Kier, P. G. Seybold, and C.-K. Cheng, *Modeling Chemical Systems Using Cellular Automata*, Springer, New York, NY, USA, 2005.

[60] M. Schnegg and D. Stauffer, "Dynamics of networks and opinions," *International Journal of Bifurcation and Chaos*, vol. 17, no. 7, pp. 2399–2409, 2007.

[61] T. C. Shelling, "Dynamic models of segregation," *Journal of Mathematical Sociology*, vol. 1, pp. 143–186, 1971.

[62] M. Rickert, K. Nagel, M. Schreckenberg, and A. Latour, "Two lane traffic simulations using cellular automata," *Physica A*, vol. 231, no. 4, pp. 534–550, 1996.

[63] A. Dorin, "Boolean networks for the generation of rhythmic structure," in *Proceedings of the Australian Computer Music Conference*, pp. 38–45, 2000.

[64] J. A. Aledo, S. Martínez, and J. C. Valverde, "Parallel dynamical systems over special digraph classes," *International Journal of Computer Mathematics*, vol. 90, no. 10, pp. 2039–2048, 2013.

[65] J. A. Aledo, S. Martínez, and J. C. Valverde, "Graph dynamical systems with general boolean states," *Applied Mathematics & Information Sciences*, vol. 9, no. 4, pp. 1–6, 2015.

[66] B. Derrida and Y. Pomeau, "Randomnetworks of automata: a simple annealed approximation," *Europhysics Letters*, vol. 1, no. 2, pp. 45–49, 1986.

[67] W. Y. C. Chen, X. Li, and J. Zheng, "Matrix method for linear sequential dynamical systems on digraphs," *Applied Mathematics and Computation*, vol. 160, no. 1, pp. 197–212, 2005.

[68] O. Colón-Reyes, R. Laubenbacher, and B. Pareigis, "Boolean monomial dynamical systems," *Annals of Combinatorics*, vol. 8, no. 4, pp. 425–439, 2004.

[69] E. A. Bender and S. G. Williamson, *A Short Course in Discrete Mathematics*, Dover Publications, New York, NY, USA, 2005.

[70] K. Morita, "Reversible cellular automata," *Journal of Information Processing*, vol. 35, pp. 315–321, 1994.

[71] T. Toffoli and N. H. Margolus, "Invertible cellular automata: a review," *Physica D: Nonlinear Phenomena*, vol. 45, no. 1–3, pp. 229–253, 1990.

[72] S. Wiggins, *Introduction to Applied Nonlinear Systems and Chaos*, vol. 2 of *Texts in Applied Mathematics*, Springer, New York, NY, USA, 1990.

[73] R. A. Hernández Toledo, "Linear finite dynamical systems," *Communications in Algebra*, vol. 33, no. 9, pp. 2977–2989, 2005.

[74] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Reading, Mass, USA, 1979.

[75] M. Sipser, *Introduction to the Theory of Computation*, PWS Publishing Company, Boston, Mass, USA, 1997.

[76] J. L. Guirao, F. L. Pelayo, and J. C. Valverde, "Modeling the dynamics of concurrent computing systems," *Computers & Mathematics with Applications*, vol. 61, no. 5, pp. 1402–1406, 2011.

[77] F. L. Pelayo and J. C. Valverde, "Notes on modeling the dynamics of concurrent computing systems," *Computers & Mathematics with Applications*, vol. 64, no. 4, pp. 661–663, 2012.

[78] W. Reisig and G. Rozenberg, *Lectures on Petri Nets I: Basic Models: Advances in Petri Nets*, vol. 1491 of *Lecture Notes in Computer Science*, Springer, New York, NY, USA, 1998.

[79] E. F. Moore, "Machine models of self-reproduction," in *Mathematical Problems in the Biological Sciences*, vol. 14 of *Proceedings of Symposia in Applied Mathematics*, pp. 17–33, 1962.

[80] T. W. Hungerford, *Algebra*, vol. 73 of *Graduate Texts in Mathematics*, Springer, New York, NY, USA, 1974.

[81] M. Gardner, "On cellular automat, self-reproduction, the Garden of Eden and the game life," *Scientic American*, vol. 224, pp. 112–117, 1971.

[82] T. Toffoli, "Cellular automata as an alternative to (rather than an approximation of) differential equations in modeling physics," *Physica D*, vol. 10, no. 1-2, pp. 117–127, 1984.

[83] T. Toffoli and N. Margolus, *Cellular Automata Machines*, MIT Press, Cambridge, Mass, USA, 1987.

[84] H. J. Blok and B. Bergersen, "Synchronous versus asynchronous updating in the 'game of Life'," *Physical Review E: Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, vol. 59, no. 4, pp. 3876–3879, 1999.

[85] T. Mihaela, T. Matache, and J. Heidel, "Asynchronous random Boolean network model based on elementary cellular automata rule 126," *Physical Review E*, vol. 71, no. 2, Article ID 026232, pp. 1–13, 2005.

[86] B. Schönfisch and A. de Roos, "Synchronous and asynchronous updating in cellular automata," *BioSystems*, vol. 51, no. 3, pp. 123–143, 1999.

[87] R. V. Sole, B. Luque, and S. A. Kauffman, "Phase transitions in random networks with multiple states," Tech. Rep. 00-02-011, Santa Fe Institute, 2000.

[88] B. Luque and F. J. Ballesteros, "Random walk networks," *Physica A*, vol. 342, no. 1-2, pp. 207–213, 2004.

[89] I. Shmulevich and E. R. Dougherty, *Probabilistic Boolean Networks: The Modeling and Control of Gene Regulatory Networks*, SIAM, Philadelphia, Pa, USA, 2010.

[90] J. A. Aledo, S. Martínez, and J. C. Valverde, "Updating method for the computation of orbits in parallel and sequential dynamical systems," *International Journal of Computer Mathematics*, vol. 90, no. 9, pp. 1796–1808, 2013.

[91] R. Laubenbacher and B. Pareigis, "Equivalence relations on finite dynamical systems," *Advances in Applied Mathematics*, vol. 26, no. 3, pp. 237–251, 2001.