ON MODELING LARGE-SCALE MULTI-AGENT SYSTEMS WITH PARALLEL, SEQUENTIAL AND GENUINELY ASYNCHRONOUS CELLULAR AUTOMATA*

Predrag T. Tošić

Department of Computer Science, University of Houston
Houston, Texas, USA
ptosic@uh.edu

(Received April 12, 2011)

We study certain types of Cellular Automata (CA) viewed as an abstraction of large-scale Multi-Agent Systems (MAS). We argue that the classical CA model needs to be modified in several important respects, in order to become a relevant and sufficiently general model for the large-scale MAS, and so that thus generalized model can capture many important MAS properties at the level of *agent ensembles* and their long-term *collective behavior* patterns. We specifically focus on the issue of inter-agent communication in CA, and propose *sequential cellular automata* (SCA) as the first step, and *genuinely* Asynchronous Cellular Automata (ACA) as the ultimate deterministic CA-based abstract models for large-scale MAS made of simple reactive agents. We first formulate deterministic and non-deterministic versions of sequential CA, and then summarize some interesting *configuration space properties* (*i.e.*, possible behaviors) of a restricted class of sequential CA. In particular, we compare and contrast those properties of sequential CA with the corresponding properties of the classical (that is, parallel and perfectly synchronous) CA with the same restricted class of update rules. We analytically demonstrate failure of the studied sequential CA models to simulate all possible behaviors of perfectly synchronous parallel CA, even for a very restricted class of non-linear totalistic node update rules. The lesson learned is that the *interleaving semantics of concurrency*, when applied to sequential CA, is not refined enough to adequately capture the perfect synchrony of parallel CA updates. Last but not least, we outline what would be an appropriate CA-like abstraction for large-scale distributed computing insofar as the inter-agent communication model is concerned, and in that context we propose *genuinely asynchronous* CA.

DOI:10.5506/APhysPolBSupp.4.217
PACS numbers: 05.70.Jk, 89.20.Ff, 89.75.Fb

* Presented at the 2nd Summer Solstice International Conference on Discrete Models of Complex Systems, Nancy, France, June 16–18, 2010.

1. Introduction and motivation: CA models for large-scale multi-agent systems

Multi-Agent Systems (MAS) are technical, social, socio-technical, biological or other decentralized systems composed of several (two or more) agents that are capable of *mutual interaction*. The interaction among agents can take various forms, from communication via exchanging messages (for example, between two software or robotic agents) to producing changes in the agents' common environment (for example, stigmergy phenomena among social insects). The agents in MAS are usually *autonomous* entities. An agent is said to be autonomous if it exercises some degree of control over both its *internal state* and its (externally observable) *behavior* — as contrasted to being entirely controlled from the outside. Examples of such autonomously acting (artificial) agents include various types of software agents, robots, smart sensors and unmanned vehicles that are not remotely controlled. Human decision-makers can also be considered autonomous agents in various contexts. In particular, various kinds of human organizations, as well as the society in general, can be considered examples of multi-agent systems.

It is well-known that MAS can manifest self-organization and complex collective behaviors even when the individual autonomous behaviors of all individual agents, as well as the interaction patterns among those agents, are quite simple. Emerging collective dynamics of *large-scale* MAS (those typically made of anywhere from hundreds to millions or more of autonomously executing agents, depending on the context) has been an active area of research in artificial intelligence and artificial life (*e.g.*, [1, 2, 3]).

MAS are usually viewed as being at the intersection of artificial intelligence on one, and distributed computing and communication systems, on the other hand. Hence, MAS research heavily draws on the existing theories, tools and methodologies from both AI and distributed computing. What we would like to contribute to the more thorough understanding and better design of large-scale MAS are some ideas, paradigms and tools from another scientific discipline, namely, *complex dynamical systems* [4, 5, 6, 3]. Among many mathematical models of complex systems, the one class that we find particularly useful for addressing many fundamental issues in parallel and distributed computing in general, and in large-scale multi-agent systems in particular, are the classical cellular automata and some of their graph or network automata extensions and variants. We find cellular and network automata models to be particularly suitable as relatively simple but quite useful and mathematically elegant abstractions of large-scale MAS when the main interest is in *collective dynamics* of large agent ensembles, as opposed to internal deliberations of individual agents [4, 3].

Cellular Automata (CA) were originally introduced as a mathematical model of biological systems capable of self-reproduction (see Section 2 and references therein). CA have been extensively studied in many different domains, especially in the context of modeling and simulation of complex physical, biological, social and socio-technical systems and their dynamics. However, CA have also been viewed as an abstraction of massively parallel computers. While most of the previous research in computer science on CA and similar models have used those models as abstractions for parallel *hardware architectures*, we use these dynamical system models as an abstraction for *open distributed environments* at the *software* level [7,4]. More precisely, we view CA-based models as an abstraction for *autonomously executing local processes* that are reactive, persistent, and coupled to and interacting with one another and possibly also with other aspects of their environment. Even when these individual processes are rather simple, their mutual interaction and synergy may potentially yield a highly complex and difficult to predict *long-term global* behavior. This property that the behavior of the "whole" (the entire system) cannot be easily deduced from the simple and well-understood behaviors of the "pieces" (individual components), is a hallmark property of both non-linear complex dynamical systems in physics and open distributed systems in computer science [4].

What are, then, the important properties of large-scale distributed computational and communication systems in general, and MAS in particular, that can be adequately captured by the classical CA and CA-like models? Let us consider a cellular automaton from a MAS perspective. Studying global dynamics of a CA then translates into an exploration of the global behavior of a multi-agent system when *(i)* the individual agent behaviors are fixed, *(ii)* the pattern of multi-agent interaction ("network topology") is fixed, and *(iii)* both the individual agent behaviors and the interaction patterns among the agents are highly regular and uniform (*i.e.*, *homogeneous*) across the entire system. In particular, CA and other related models capture the fundamental MAS properties of *locality* of interaction among the agents, and the bounded speeds of information and impact propagation [4,5].

Several modifications of the basic CA model along different dimensions can be readily argued to provide appropriate abstractions for the *large-scale multi-agent systems*. We have identified the following four as the most important [5]:

- *heterogeneity* of the network or graph automata models in terms of *(i)* the individual agent behaviors and *(ii)* the inter-agent interaction pattern, in contrast to the strict *homogeneity* of the classical CA in both these respects;

- *model of inter-agent communication* insofar as whether the agents locally update their states synchronously or asynchronously, and whether they interact (communicate) with one another synchronously or asynchronously;
- *adaptability* of the individual agents, *i.e.*, are these agents capable of *dynamically changing their behavior* via, *e.g.*, reinforcement learning, or are their individual behaviors *fixed* once the conditions of the environment and the current state of the agent are specified;
- *dynamic changes* of the MAS network topology, that would be captured by allowing the underlying cellular space of a CA to change as a function of time.

In this paper, we focus on the second dimension in the above list. More specifically, some implications of the *perfect synchrony* of the parallel CA node updates will be identified, and CA behaviors studied once this physically unrealistic assumption is dropped.

Namely, classical CA are characterized by the *perfect synchrony* of the parallel node updates. This perfect synchrony implies, in effect, *logical simultaneity*, and is hard to justify on either physics or computer science grounds [4, 8]. By allowing the nodes to update one at a time, one arrives at a sequential version of CA, called Sequential Cellular Automata (SCA). However, these sequential cellular (and more general graph) automata models, while more realistic in most domains than their synchronous parallel counterparts, still fall short of being an appropriate abstract model for large-scale distributed computation, due to the underlying assumptions of a global clock, and therefore *communication synchrony* [8, 9]. That is, the local computations of agents are indeed asynchronous, but the inter-agent *interaction* is still implicitly assumed synchronized. Therefore, the natural next step is to study properties of what we call *genuinely asynchronous* cellular or graph automata, where no synchrony is assumed when it comes to either local computation or agent-to-agent communication.

2. Cellular automata basics

Cellular Automata (CA) were originally introduced as an abstract mathematical model that can capture the behavior of biological systems capable of self-reproduction [10]. Subsequently, CA have been extensively studied in a great variety of domains; they hold a particularly prominent place in modeling and simulation of complex physical, biological or social systems and their emerging behavior and collective dynamics (*e.g.*, [11, 12, 13, 14, 15, 16, 17]).

However, CA can also be considered an abstraction of *massively parallel computers* [18, 7]. In our work on various communication models for CA, we pose, and partly answer, some fundamental questions regarding the nature

of CA *parallelism*, *i.e.*, the concurrency of the classical CA computation. To keep the underlying mathematics manageable, the analysis is done in the context of fairly simple but non-trivial totalistic update rules, namely, the *simple threshold* CA. We recall that CA are often viewed as a computational model of *fine-grain parallelism* [18, 7], in that the elementary operations executed at each node are rather simple and hence comparable to the basic operations performed by the computer hardware. However, due to the interaction and synergy among a typically great number of these nodes, many CA are capable of highly complex behaviors (or, equivalently, computations). In a parallel CA, all the nodes execute their operations *logically simultaneously*: the state of node x_i at time step $t+1$ is a function of the states of node x_i itself, and a set of its pre-specified neighbors at time t.

We consider in the sequel a sequential version of CA, abbreviated SCA, and compare it with the classical, *parallel* CA. In particular, we show that there are 1-D CA with very simple node state update rules that cannot be simulated by any comparable SCA, irrespective of the node update ordering. It will then follow that the granularity of the basic CA operations, insofar as the ability to simulate their concurrent computation via appropriate nondeterministic *sequential interleavings* [19, 20] of these basic operations, turns out not to be *fine enough* [8, 9].

We will also share some thoughts on how to extend our results, and, in particular, we try to motivate the study of *genuinely asynchronous cellular automata*, where asynchrony applies not only to the local computations at individual nodes, but also to *communication* among different nodes (via "shared variables" stored as the respective nodes' states) [4, 9].

An example of asynchrony in the local node updates (*i.e.*, asynchronous computation at different "processors") is the case when, for instance, the individual nodes update one at a time, according to some random order. This is a kind of asynchrony that is commonplace in the existing literature; see, *e.g.*, [2, 21, 22]. It is important to understand, however, that even in case of what is referred to as *Asynchronous Cellular Automata* (ACA) in much of the existing literature, the term *asynchrony* there applies to local updates (*i.e.*, computations) *only*, but not necessarily to *communication* between the nodes, since a tacit assumption of the globally accessible global clock still holds. (However, for a rare exception to this general rule, see the second asynchronous model discussed in [22].) We prefer to refer to this kind of (weakly asynchronous) (A)CA as *Sequential Cellular Automata*, and, in this work, consistently keep the term ACA for those CA that do not have a global clock (see Section 5).

We remark that we use the terms *parallel* and *concurrent* as synonyms throughout the paper. This need not be the most standard convention among the researchers of programming languages and semantic models of

concurrency (*e.g.*, [23, 24, 25]); however, we are certainly not alone in not making the distinction between the two notions (see, *e.g.*, discussion in [24]). Moreover, by a *parallel* (equivalently, *concurrent*) *computation*, we mean actions of several processing units that are carried out *logically* (if not necessarily *physically*) *simultaneously*.

Definition 2.1 *A Cellular Space, Γ, is an ordered pair (G, Q), where*
- *G is a regular undirected Cayley graph that may be finite or infinite, with each node labeled with a distinct integer; and*
- *Q is a finite set of states that has at least two elements, one of which being the special* quiescent state*, denoted by* 0.

We denote the set of integer labels of the nodes in Γ by L. That is, L may be equal to, or be a proper subset of, the set of all integers.

Definition 2.2 *A Cellular Automaton $\boldsymbol{A}$ is an ordered triple (Γ, N, M), where*
- *Γ is a cellular space;*
- *N is a fundamental neighborhood; and*
- *M is a finite state machine whose input alphabet is $Q^{|N|}$, and the local transition function (update rule) for each node is of the form δ: $Q^{|N|+1} \rightarrow Q$ for CA with memory, and δ: $Q^{|N|} \rightarrow Q$ for memoryless CA.*

The fundamental neighborhood N specifies what nearby nodes provide inputs to the update rule of a given node. In classical CA, Γ is a regular graph that locally "looks the same everywhere": local neighborhood N is the same for each node in Γ. The local transition rule δ specifies how each node updates its state (that is, value), based on its current state (value), and the current states of its neighbors in N. By composing together the application of the local transition rule to each of the CAs nodes, we obtain *the global map* on the set of global configurations of a CA.

Definition 2.3 *A Sequential Cellular Automaton (SCA) $\boldsymbol{S}$ is an ordered quadruple (Γ, N, M, s), where Γ, N and M are as in Definition 2.2, and s is an arbitrary sequence, finite or infinite, all of whose elements are drawn from the set L of integers used in labeling the vertices of Γ. The sequence s is specifying the sequential ordering according to which an SCA's nodes update their states, one at a time.*

However, when comparing and contrasting the concurrent CA with their sequential counterparts, rather than making a comparison between a given CA with a *particular* SCA (that is, a corresponding SCA with some particular choice of the update sequence s), we compare the parallel CA computations with the computations of the corresponding SCA for *all* possible sequences of node updates.

Definition 2.4 *A Nondeterministic Interleavings Cellular Automaton (NICA)* $\boldsymbol{I}$ *is defined to be the union of all sequential automata* $\boldsymbol{S}$ *whose first three components,* Γ, N *and* M *are fixed. That is,* $I = \cup_s(\Gamma, N, M, s)$*, where the meanings of* Γ, N, M*, and* s *are as before, and the union is taken over all infinite sequences* $s : \{1, 2, 3, \dots\} \to L$*, where* L *is the set of (integer) labels of the nodes in* Γ*.*

We next introduce some concepts and terminology from physics that is useful for characterizing *all possible computations* of various parallel and sequential CA. In particular, we adopt a *(discrete) dynamical system* view of CA. A *phase space* of a dynamical system is a directed graph where the vertices are the *global configurations* (or *global states*) of the system, and directed edges correspond to direct transitions from one global state to another. One can now define the fundamental, qualitatively distinct types of global configurations that a CA can find itself in. These different types of global configurations relate to key properties of *asymptotic dynamics* of CA (or other formal models when viewed as discrete dynamical systems). We are therefore interested in studying configuration space properties of parallel, sequential and asynchronous CA as they capture qualitatively distinct types of possible emerging dynamics in systems abstracted as those various types of CA [4, 3].

We first define the fundamental types of dynamical system configurations for the parallel CA, and then discuss how these definitions need to be modified for SCA and NICA. The classification below is based on answering the following question: starting from a given global CA configuration, can the automaton return to that same configuration after a finite number of parallel computational steps?

Definition 2.5 *A fixed point (FP) is a configuration in the phase space of a CA such that, once the CA reaches this configuration, it stays there forever. A cycle configuration (CC) is a state that, once reached, will be revisited infinitely often with a fixed, finite temporal period of 2 or greater. A transient configuration (TC) is a state that, once reached, is never going to be revisited again.*

In particular, a FP is a special, degenerate case of a recurrent state with period 1. Due to deterministic evolution, any configuration of a classical, parallel CA is either a FP, a "proper" CC, or a TC. Throughout, we shall make a clear distinction between FPs and proper CCs.

On the other hand, if one considers SCA so that *arbitrary* node update orderings are permitted, then, given the underlying cellular space and the local update rule, the resulting phase space configurations, due to nondeterminism that results from different choices of possible sequences of node

updates are more complicated. In a particular SCA, a cycle configuration is any configuration revisited infinitely often — but the period between different consecutive visits, assuming an arbitrary sequence s of node updates, need not be fixed. We call a global configuration that is revisited only finitely many times (under a given ordering s) *quasi-cyclic*. Similarly, a *quasi-fixed point* is an SCA configuration such that, once the SCA's evolution reaches this configuration, it stays there "for a while", and then leaves. For example, a configuration of an SCA can simultaneously be both an FP and a quasi-CC, or both a quasi-FP and a CC [8]. We call a configuration C of a NICA a *(weak) fixed point* if there exists an infinite sequence of node updates s such that C is a FP in the usual sense when the NICA's nodes update according to the ordering s. A *strong fixed point* of a NICA is a configuration that is fixed (stable) with respect to *all* possible sequences of node updates. A NICA configuration C' is a cycle state, if there exists an infinite sequence of node updates s' such that, if a NICA nodes update according to s', then C' is a cycle state of period 2 or greater in the usual sense (see Definition 2.5). In particular, a single configuration of a given NICA can simultaneously be a weak FP, a CC and a TC; see [8] for an example.

3. Related work

In this section, we first briefly summarize the prior art on abstracting collective dynamics and emerging behavior of autonomous agents and multi-agent systems as appropriate models of *coupled automata* or *communicating finite state machines* (CFSMs) (see, *e.g.*, [26, 4]); we then relate those modeling efforts to our main research objectives in this paper. We, however, do not pursue a broader survey of the literature on various computational and/or dynamical aspects of CA; some of the seminal work on various computational and dynamical properties of CA and related models can be found in the papers briefly surveyed in the previous section, and references found in those papers.

It has been argued that there is no such a thing as *disembodied mind* or *disembodied intelligence*. In particular, all intelligent agents, whether biological or artificial, are embedded in an appropriate environment, be it physical or virtual, and act upon that environment [27, 28]. Any non-trivial autonomous agent in a multi-agent system always has an aspect of the environment external to it, yet relevant to its behavior: that aspect are the other agents in the system. There may be also other aspects of the external environment relevant to an agent's objectives and, in general, potentially impacting the agent's behavior. Some examples of other relevant aspects of the environment may include tasks, external resources, physical or other types of obstacles interfering with the agent's pursuit of its objec-

tives, and so on. Hence, over the past 10–15 years, many directions of AI and MAS research have begun treating and, in particular, explicitly *modeling* the agents' environments as the first-class entities; some prominent examples include [1, 29, 30, 31].

That the *coupling* between an embedded, or *situated*, agent and its environment can be abstracted via the *coupled automata* formal model was first raised (as far as we know) in the seminal work by Kaelbling on situated agents [26]. In that paper, a *single* autonomous agent embedded in a dynamic environment is studied. Coupled or communicating finite automata models can, however, clearly be extended to capture multiple agents interacting with each other and with their environment. This modeling framework is generally applicable when an agent's environment is made entirely of other agents, as well as when both other agents and non-agent entities constitute the relevant aspects of an agent's environment [4]. Classical CA (with any of the communication models discussed in this paper), in which each agent is abstracted as the same kind of a finite state machine, are an appropriate abstraction of a MAS made of *identical* reactive agents, and where all relevant aspects of an agent's environment are adequately captured by appropriately defining the local interactions of that agent with its neighboring agents. More general graph and network automata, where different nodes correspond, in general, to different finite state machines, can be viewed as appropriate abstractions of *heterogeneous* MAS where not all agents necessarily behave the same way [4,3]. Alternatively, in such more general network automata based MAS models, one kind of update rules may be capturing individual behaviors of agents, and another kind of rules, the behavior of the outside environment. Indeed, motivated by some classical problems in robotics, the original coupled automata model of situated agency in [26] intends to capture the interaction and mutual impact between an autonomous agent and its (non-agent) external environment.

Communication models in cellular and network automata and, in particular, the issue of (a)synchrony of the node updates, have been studied in the CA and complex systems literature since at least the 1980s. It was originally observed in [22] that certain properties of the parallel CA dynamics are essentially peculiarities of, generally speaking, a rather unrealistic assumption of the perfect synchrony of parallel node updates. The authors of [22] propose two "asynchronous" models; the second one, based on different nodes updating according to different local clocks, is the closest to our proposed model discussed in Section 5 below. The impact of different communication models on the emerging dynamics of CA and other, similar models of local interactions has been explicitly studied in several other papers, such as [32, 2, 21]. We remark that our own early work on parallel and sequential CA was prompted by the observation that the *temporal cycles* in parallel

CA with the *Majority* update rule are indeed an idiosyncrasy of the perfect synchrony assumption, that cannot be reproduced by any conceivable sequential ordering of node updates (see [4, 8] and the summary of pertinent analytical results in the next section of the present paper).

4. On temporal cycles in parallel and sequential simple threshold CA

We now compare and contrast classical, concurrent and perfectly synchronous CA with their sequential counterparts, SCA and NICA, in the context of the simplest non-linear local update rules possible, namely, *simple threshold functions*.

We first formally define *(simple) linear threshold functions* and the corresponding types of (S)CA.

Definition 4.1 *A Boolean-valued linear threshold function of m inputs, $x_1, \ldots, x_m$, is any function of the form*

$$f(x_1, ..., x_m) = \begin{cases} 1\,, & \text{if } \sum_i w_i \times x_i \geq \theta \\ 0\,, & \text{otherwise} \end{cases} \quad , \tag{4.1}$$

where θ is an appropriate threshold constant, and $w_1, \ldots, w_m$ are arbitrary (but fixed) non-negative real numbers[1] *called weights.*

A threshold automaton (threshold (S)CA) is a cellular automaton where δ is a Boolean-valued linear threshold function.

Therefore, given an integer k, a *k-threshold function*, in general, is any function of the form as in Def. 4.1 with $\theta = k$ and an appropriate choice of weights w_i, $i = 1, \ldots, m$. We consider in this paper *monotonically non-decreasing* Boolean threshold functions only; this restriction, in particular, implies that all weights w_i are non-negative. We also additionally assume δ to be a *symmetric function* of all of its inputs, thereby making the CA, SCA and NICA models with such update rules *totalistic* [14, 15]. That is, the (S)CA and NICA we analyze have *symmetric, monotone Boolean functions* for their local update rules. We call such functions *simple threshold functions*, and we refer to the (S)CA and NICA with simple threshold node update rules as to *simple threshold* (S)CA/NICA.

[1] In general, w_i can be both positive and negative. This is esp. common in the neural networks literature, where negative weights w_i indicate an *inhibitory effect* of, *e.g.*, one neuron on the firings of another, near-by neuron. In most studies of discrete dynamical systems, however, the weights w_i are required to be non-negative — that is, only *excitatory effects* of a node on its neighbors are allowed; see, *e.g.*, [33,34,14,15].

Definition 4.2 *A simple threshold (S)CA or NICA is a cellular automaton whose local update rule δ is a monotone symmetric Boolean threshold function.*

Throughout, whenever we say *threshold cellular automaton* we shall mean a *simple* threshold cellular automaton, unless explicitly stated otherwise. That is, the 1-D threshold (S)CA studied in the sequel have the node update functions of the general form

$$\delta(x_{i-r}, \ldots, x_i, \ldots, x_{i+r}) = \begin{cases} 1\,, & \text{if } \sum_{j=-r}^{r} x_{i+j} \geq k \\ 0\,, & \text{otherwise} \end{cases}\,, \qquad (4.2)$$

where k is a fixed integer from the range $0, 1, \ldots, 2r+1, 2r+2$. For example, if the CA rule radius is $r = 2$, and if $k = 2$, then the k-threshold (S)CA on a specified number of nodes is just the (S)CA with the node update rule $\delta =$ *at least 2 out of 5*: the update rule evaluates to 1 if and only if at least two out of five of its inputs are currently equal to 1. Arguably the most interesting simple threshold update rule is the *Majority* function, which we will abbreviate as MAJ. In a CA with $\delta =$ MAJ, a node updates to 1 if and only if the simple majority of its current inputs are 1.

The results below hold for two-way infinite 1-D (S)CA, as well as for finite (S)CA with the circular boundary conditions — that is, for the (S)CA whose cellular spaces are finite rings.

Theorem 4.1 [6] *For any Simple Threshold Boolean 1-D Sequential CA $\boldsymbol{A}$ with rule radius $r = 1$, and any sequence s of the node updates, the phase space $\boldsymbol{PS(A)}$ of the cellular automaton $\boldsymbol{A}$ is temporal cycle-free. In contrast, parallel simple threshold CA with $\delta =$ MAJ and $r = 1$ do have temporal two-cycles in their configuration spaces.*

More generally, for any underlying cellular space Γ that is a (finite or infinite) *bipartite graph*, the corresponding (non-trivial) CA with $\delta =$ MAJ have temporal two-cycles. We remark that bipartiteness of Γ is sufficient, but not necessary, for the existence of two-cycles in this setting [9].

It turns out that the two-cycles in the *PS* of concurrent CA with $\delta =$ MAJ are actually *the only type of (proper) temporal cycles* such cellular automata can have. Indeed, for any *simple threshold update rule* δ, and any *finite* regular Cayley graph as the underlying cellular space, the following general result holds [11, 6, 9]:

Proposition 4.1 *Let a classical, parallel simple threshold CA $\boldsymbol{A} = (\Gamma, N, M)$ be given, where Γ is any finite cellular space, and let this cellular automaton's global map be denoted by F. Then for all configurations $C \in PS(A)$, there exists a finite time step $t \geq 0$ such that $F^{t+2}(C) = F^t(C)$.*

In particular, this result implies that for any *finite* simple threshold CA, and for any starting configuration C_0, there are *only two* possible kinds of orbits: the computation either converges to a fixed point configuration after finitely many steps, or else it eventually arrives at a two-cycle.

It is almost immediate that, if we allow the underlying cellular space Γ to be infinite, if the computation from a given starting configuration converges after any finite number of steps at all, it will have to converge either to a fixed point or a two-cycle (but never to a cycle of, say, period three — or, for that matter, of any other finite period). This property also extends to finite and infinite SCA, provided that we reasonably define what is meant by a *single computational step* in a situation where the nodes update one at a time. The simplest such notion is that of a single node updating its state. (One undesirable consequence is, that a single parallel step of a classical CA defined on an infinite underlying cellular space Γ includes an infinite amount of sequential computation and, in particular, infinitely many elementary sequential steps.)

Additionally, in order to ensure some sort of convergence of an arbitrary SCA (especially when the underlying Γ is infinite), and, more generally, to ensure that *all the nodes* get a chance to update their states, an appropriate condition that guarantees *fairness* needs to be specified. That is, an appropriate restriction on the allowable sequences s of node updates is required. As a first step toward that end, we shall allow only *infinite* sequences s of node updates through the rest of the paper. For SCA defined on finite cellular spaces, one sufficient fairness condition is to impose a fixed upper bound on the number of sequential steps before any given node gets its "turn" to update again. This is the simplest generalization of the fixed permutation assumption made in the work on sequential and synchronous dynamical systems; see, *e.g.*, [33, 34, 35, 36, 37, 5, 38].

In the infinite SCA case, on the other hand, the issue of fairness is nontrivial, and some form of *dove-tailing* of sequential individual node updates may need to be imposed. We shall require from the sequences s of node updates of the SCA and NICA threshold automata to be fair in a simple sense defined below [8]:

Definition 4.3 *An infinite sequence $s : N \to L$ is fair if (i) the domain L is finite or countably infinite, and (ii) every element $x \in L$ appears infinitely often in the sequence of values $s(1) = s_1, s(2) = s_2, s(3) = s_3, \ldots$*

We have now set the stage for the following generalization of *Proposition 4.1* to both finite and infinite 1-D CA and SCA:

Proposition 4.2 *Let a parallel or sequential (S)CA be defined over a finite or infinite 1-D cellular space (that is, a line or a ring), with a finite rule radius* $r \geq 1$. *Let this (S)CA's local update rule be a simple threshold function. Let us also assume, in the sequential cases, that the fairness condition from Definition 4.3 holds.*

Then for any starting configuration $C_0 \in PS(A)$ *whatsoever, and any finite subconfiguration* $C \subseteq C_0$, *there exists a time step* $t \geq 0$ *such that*

$$F^{t+2}(C) = F^t(C)\,, \tag{4.3}$$

where, in the case of fair SCA, *the Eq. (4.3) can be replaced with*

$$F^{t+1}(C) = F^t(C)\,. \tag{4.4}$$

For the special case of $\delta =$ MAJ (S)CA, a computation starting from any *finitely supported* initial configuration[2] necessarily (and relatively *quickly* [8,9]) converges to either a FP or a two-cycle [11]:

Proposition 4.3 *Let the assumptions from Proposition 4.2 hold, and let the underlying threshold rule be* $\delta =$ MAJ. *Then for all configurations* $C \in PS(A)$ *whatsoever in the finite cases, and for all configurations* $C \in PS(A)$ *such that* C *has a* finite support *when* $\Gamma(A)$ *is infinite, there exists a finite time step* $t \geq 0$ *such that* $F^{t+2}(C) = F^t(C)$. *Moreover, in the sequential cases with* fair *update sequences, there exists a finite* $t \geq 0$ *such that* $F^{t+1}(C) = F^t(C)$.

Furthermore, if *arbitrary* infinite initial configurations are allowed in Propositions 4.2–4.3, and the dynamic evolution of such global configurations is monitored, then the only additional possibility is that the particular (S)CA computation fails to finitely converge altogether.

To summarize, if the computation of a SCA starting from some configuration C converges at all (that is, to *any type of* finite temporal cycle), it actually has to converge to a fixed point.

Full proofs of the above two propositions can be found in [39]; more detailed discussion of the claims in the propositions, and related properties of parallel and sequential Threshold CA over finite and infinite cellular spaces, can be found in our prior work [8,6,9].

For completeness, we state one more result that provides a stark contrast of possible dynamics between parallel and sequential Simple Threshold CA. This result was used in [39] to establish Proposition 4.3 above.

[2] Also sometimes called *compact support;* see, *e.g.*, [11]. A global configuration of a cellular automaton defined over an infinite cellular space Γ is said to be *compactly supported* if all except for at most finitely many of the nodes are *quiescent* (*i.e.*, in state 0) in that configuration.

Theorem 4.2 [9,39] *The following dichotomy holds:*

(i) All 1-D (parallel) CA with any odd $r \geq 1$*, the local rule* $\delta = \mathrm{MAJ}$*, and cellular space* Γ *that is either a finite ring with an even number of nodes or a two-way infinite line, have finite cycles in their phase spaces. The same holds for* arbitrary *(even or odd)* $r \geq 1$ *provided that* Γ *is either a finite ring with a number of nodes divisible by* $2r$*, or a two-way infinite line*[3]*.*

(ii) Any 1-D SCA with any *simple threshold Boolean update rule* δ*, any finite* $r \geq 1$*, defined over any (finite or infinite) 1-D cellular space, and for* an arbitrary *sequence* s *(finite or infinite, fair or unfair) as the node update ordering, has a cycle-free phase space.*

To summarize, simple threshold CA, depending on the starting configuration, may converge to a fixed point or a temporal two-cycle; in particular, they may end up "looping" in *finite but non-trivial* temporal cycles. In contrast, the corresponding classes of SCA (and therefore NICA) can never cycle. Moreover, given an arbitrary sequence of node updates of a finite threshold SCA, if this sequence satisfies an appropriate *fairness condition*, then the evolution of such a threshold SCA A is guaranteed to converge to a fixed-point (sub)configuration on any finite subset of the nodes in $\Gamma(A)$.

The temporal cycle-freeness of the threshold SCA and NICA holds irrespective of the choice of sequential update ordering; furthermore, extending to infinite SCA, temporal cycles cannot be obtained even "in the limit", that is, via infinitely long computations obtained by allowing arbitrary infinite sequences of individual node updates. Hence, we conclude that no choice of a "sequential interleaving" can capture the perfectly synchronous parallel computation of parallel simple threshold CA. Consequently, the *interleaving semantics* [19,20,23,40] of NICA fails to capture the synchronous parallel behavior of classical CA even for this, simplest non-linear class of *totalistic* [8,14,15] CA update rules.

5. Future directions for modeling MAS by cellular automata: Genuinely Asynchronous CA

Among other things, the results in Section 4 show that, even for the very simplest non-linear and non-affine totalistic cellular automata, *nondeterministic interleavings* dramatically fail to capture the perfectly synchronous parallelism of classical CA. In the actual engineering, physical or biological systems, however, such perfect synchrony is usually hard to justify.

[3] We note that there are also certain CA defined over finite rings and with even $r \geq 2$ such that the number of nodes in these rings is not divisible by $2r$ yet temporal two-cycles exist. However, a more detailed discussion on what properties the number of nodes in such CA has to satisfy would be required; we leave further discussion out for clarity reasons, since those details are not of major importance for our core objectives in this paper.

In particular, when CA are applied to modeling various complex physical or biological phenomena (be those the crystal growth, the forest fire propagation, the information or gossip diffusion in a population, or the signal propagation in an organism's neural system), one wants to chiefly focus on the underlying CA behaviors that are, in some sense, *dynamically robust*. From this standpoint, temporal cycles in the parallel threshold CA are, indeed, an idiosyncrasy of the perfect synchrony, an oddity that is anything but robust. Likewise, it makes sense to focus one's qualitative study of the dynamical systems modeled by the threshold CA to those properties that are *statistically robust* [41]. It can be readily argued in a rigorous, probabilistic sense that, again, the typical, statistically robust behavior of a simple threshold CA computation is a relatively short transient chain, followed by the convergence to a stable fixed point. In particular, the non-fixed-point temporal cycles of the threshold CA not only utterly lack any non-trivial basins of attraction (in terms of the incoming transient 'tails'), but are themselves statistically negligible for all sufficiently large finite, as well as for all infinite, cellular automata [6, 9].

Our recent work on the CA-like complex system models of large-scale MAS has considered other communication models, that is, other assumptions on (a)synchrony of the CA inter-agent communication. We remark that the two particular classes of graph automata defined over arbitrary (not necessarily regular, or Cayley) *finite* graphs, namely, the sequential and synchronous dynamical systems (SDSs and SyDSs, respectively), and their various phase space properties, have been extensively studied; see, *e.g.*, [33, 34, 35, 36, 38] and references therein. It would be interesting, therefore, to consider *asynchronous cellular and graph automata*, where the nodes are not assumed any longer to update in unison and, moreover, where *no global clock is assumed* [8, 9]. We again emphasize that such automata would entail what can be viewed as *communication asynchrony*, thus going beyond the sequentiality (random or otherwise) of the node updates that has been relatively extensively studied since the 1980s.

What are, then, such *genuinely asynchronous* CA like? How do we specify the local update rules, that is, computations at different nodes, given the possible "communication delays" in what was originally a multiprocessor-like, rather than distributed system-like, parallel model? In the parallel case where a perfect communication synchrony is assumed, any given node x_i of a 1-D CA of radius $r \geq 1$ updates according to

$$x_i^{t+1} = f\left(x_i^t, x_{i_1}^t, \ldots, x_{i_{2r}}^t\right) , \tag{5.1}$$

for an appropriate local update rule $\delta = f(x_i, x_{i_1}, \ldots, x_{i_{2r}})$, whereas, in the asynchronous case, the individual nodes update according to

$$x_i^{t+1} = f\left(x_i^t, x_{i_1}^{t_1}, \ldots, x_{i_{2r}}^{t_{2r}}\right) . \tag{5.2}$$

We observe that t in Eq. (5.1) pertains to *the global time*, which of course in this case also coincides with the node x_is (and everyone else's) *local time*. However, in case of Eq. (5.2), each t_j pertains to an appropriate *local time*, in the sense that each $x_{i_j}^{t_j}$ denotes the node x_{i_j}s value that was most recently received by the node x_i.

That is, $x_{i_j}^{t_j}$ is a local view of the node x_{i_j}s state, as seen by the node x_i. Thus, the non-existence of the global clock has considerable implications. How to meaningfully relate these different local times, so that one can still mathematically analyze such ACA — yet without making the ACA description too complicated[4]? Yet, if we want to study *genuinely* asynchronous CA models (rather than arbitrary sequential models with global clocks), these changes in the definition seem unavoidable.

This *genuine* (that is, communication) asynchrony in CA (see Eq. (5.2)) can be readily interpreted in non-deterministic terms: at each time step, a particular node updates by using its own current value, and also non-deterministically choosing the current or one of the past values of its neighbors. Such a "past value" of a node x_{i_j} used by the node x_i would be only required not to be any older than that value of x_{i_j} that x_i had used as its input on its most recent local computation, *i.e.*, on the node x_is most recent previous turn to update. Hence, insofar as what are the current inputs to any given node's update function δ, there is a natural non-deterministic interpretation of the fact that different nodes have different clocks.

A systematic further qualitative and quantitative comparative study of Asynchronous CA *vs.* the sequential and parallel CA models, in our opinion, could provide valuable insights into those emerging behaviors of large-scale MAS that are primarily or even solely due to *communication delays*.

6. Summary

We propose modeling "Multi-Agent Systems" (MAS) made of embedded reactive autonomous agents and, in particular, qualitative properties of the asymptotic global dynamics of such MAS, by the cellular automata based models. In order to make these CA-based models a suitable abstraction for large-scale MAS, several basic properties of classical CA need to be reconsidered. In this paper, we focus on the CA *communication models*.

We study possible computations of a simple class of *totalistic* cellular automata when the unrealistic assumptions of *perfect synchrony* and *instantaneous unbounded parallelism* are dropped. Motivated by the notion of *sequential interleaving semantics of concurrency*, we apply this metaphor

[4] That is, while staying away from introducing explicit message *sends* and *receives*, (un)bounded buffers, and the like.

to parallel CA and propose a simple formal model of *sequential cellular automata*, SCA, and *sequential interleavings cellular automata*, NICA. In that context, we summarize our comparative study between SCA and NICA models on one hand, and parallel CA models on the other, when the node update rules are restricted to *simple threshold functions* [8, 6, 9]. It turns out that even in such very restricted scenarios, this sequential "interleaving semantics" of NICA fails to adequately capture the perfectly synchronous parallelism of classical parallel CA.

Motivated by these results on sequential and parallel threshold CA and some of their implications, we next consider what would be an appropriate communication model for the CA applicable to *large-scale distributed computing*. In particular, we propose a class of *genuinely asynchronous* cellular automata (ACA), and indicate potential benefits of a rigorous study of ACA properties. These benefits include *(i)* identifying some of the collective dynamics properties of large-scale MAS that are mainly or entirely due to communication asynchrony, *(ii)* appropriate automata-based sequential semantics models for parallel and distributed computation, and *(iii)* lower bounds on the computational (in)tractability of answering general questions about the long-term behavior of large-scale MAS; examples of such questions include those related to the stable global configuration existence, the expected or worst-case rates of convergence to such stable configurations, determining whether a given global MAS configuration is recurrent or transient, and so on.

The broader agenda behind further study of different communication models for cellular and graph automata, such as the SCA, NICA and ACA models discussed in this paper and analyzed in more detail in our related work [39, 8, 9], is to gain new insights into the emerging behavior and collective dynamics of various large-scale multi-agent systems and other distributed computational and communication infrastructures via modeling, simulating and analyzing them from a dynamical systems perspective.

REFERENCES

[1] S.A. Brueckner, H. Van Dyke Parunak, Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-03), ACM, New York, NY, USA, 2003.

[2] D. Cornforth, D.G. Green, D. Newth, *Physica D* **204**, 70 (2005).

[3] P. Tosic, G. Agha, Proceedings of the 3rd European Workshop on Multiagent Systems (EUMAS'05), Brussels, Belgium, 2005, pp. 415–426.

[4] P. Tosic, Cellular Automata for Distributed Computing: Models of Agent Interaction and Their Implications, IEEE International Conference on Systems, Man and Cybernetics (SMC'05), Waikoloa, The Big Island of Hawaii, USA 2005.

[5] P. Tosic, *Lect. Notes Comput. Sci.* **3993**, 272 (2006).

[6] P. Tosic, G. Agha, *Lect. Notes Comput. Sci.* **3305**, 861 (2004).

[7] P. Tosic, Proceedings ACM Computing Frontiers (CF'04), Ischia, Italy, 2004.

[8] P. Tosic, G. Agha, APDCM Workshop, Proceedings of the 18th IEEE International Parallel and Distributed Processing Symposium, Santa Fe, New Mexico, USA, 2004.

[9] P. Tosic, G. Agha, Proceedings of the First European Conference on Complex Systems (ECCS'05), Paris, France, 2005.

[10] J. von Neumann, *Theory of Self-Reproducing Automata*, edited and completed by A.W. Burks, Univ. of Illinois Press, Urbana 1966.

[11] E. Goles, S. Martínez, *Neural and Automata Networks: Dynamical Behavior and Applications*, Kluwer, 1990.

[12] E. Goles, S. Martínez (eds.), *Cellular Automata and Complex Systems*, Kluwer, 1999.

[13] H. Gutowitz (ed.), *Cellular Automata: Theory and Experiment*, The MIT Press/North-Holland, 1991.

[14] S. Wolfram, *Phys. Scr.* **T9**, 170 (1985).

[15] S. Wolfram (ed.), *Theory and Applications of CA*, World Scientific, Singapore 1986.

[16] S. Wolfram, *Cellular Automata and Complexity* (collected papers), Addison-Wesley, 1994.

[17] S. Wolfram, *A New Kind of Science*, Wolfram Media, Inc., 2002.

[18] M. Garzon, *Models of Massive Parallelism: Analysis of Cellular Automata and Neural Networks*, Springer, 1995.

[19] I. Czaja, R.J. van Glabbek, U. Goltz, *Lect. Notes Comput. Sci.* **609**, 89 (1992).

[20] R.J. van Glabbek, U. Goltz, *Lect. Notes Comput. Sci.* **469**, 309 (1990).

[21] N. Fatès, E. Thierry, M. Morvan, N. Schabanel, *Theor. Comput. Sci.* **362**, 1 (2006).

[22] T.E. Ingerson, R.L. Buvel, *Physica D* **10**, 59 (1984).

[23] R. Milner, *A Calculus of Communicating Systems*, Lecture Notes in Computer Science, Springer-Verlag, 1980.

[24] J.C. Reynolds, *Theories of Programming Languages*, Cambridge Univ. Press, 1998.

[25] R. Sethi, *Programming Languages: Concepts and Constructs* (2nd ed.), Addison-Wesley, 1996.

[26] L.P. Kaelbling, *ACM SIGART Bulletin* **2**, 85 (1991).

[27] R.A. Brooks, Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91), Sydney, Australia; J. Myopoulos, R. Reiter (eds.), pp. 569—595.

[28] S.J. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd edition, Prentice Hall series in AI, Pearson, 2010.

[29] L. Gardelli, M. Viroli, M. Casadei, A. Omicini, *Lect. Notes Comput. Sci.* **4389**, 254 (2007).

[30] A. Helleboogh, G. Vizzari, A. Uhrmacher, F. Michel, *Auton. Agent. Multi-Agent Syst.* **14**, 87 (2007).

[31] D. Weyns, A. Omicini, J. Odell, *Auton. Agent. Multi-Agent Syst.* **14**, 5 (2007).

[32] R. Cori, Y. Métivier, W. Zielonka, *Inform. Comput.* **106**, 159 (1993).

[33] C. Barrett *et al.*, Discrete Mathematics and Theoretical Computer Science Proceedings AA (DM-CCG), 2001, pp. 95—110.

[34] C. Barrett *et al.*, *Theor. Comput. Sci.* **295**, 41 (2003).

[35] C. Barrett, H. Mortveit, C. Reidys, *Appl. Math. Comput.* **107**, 121 (2000).

[36] C. Barrett, H. Mortveit, C. Reidys, *Appl. Math. Comput.* **122**, 325 (2001).

[37] P. Tosic, *IJFCS* **17**, 1179 (2006).

[38] P. Tosic, DMTCS Proceedings, Automata 2010 — 16th Intl. Workshop on CA and DCS, 2010, pp. 125–144.

[39] P. Tosic, *IJNCR* **1**, 66 (2010).

[40] R. Milner, *Communication and Concurrency*, Prentice-Hall, 1989.

[41] W. Ross Ashby, *Design for a Brain*, Wiley, 1960.