Cellular Automata for Distributed Computing: Models of Agent Interaction and Their Implications

PREDRAG T. TOSIC
Open Systems Laboratory, Department of Computer Science
University of Illinois at Urbana-Champaign, U.S.A.
p-tosic@cs.uiuc.edu

Abstract – *We propose cellular automata (CA) based models as a useful mathematical idealization for modeling large-scale distributed computing in general, and large-scale multi-agent systems (MAS), in particular. The classical CA need to be modified in several important respects, in order to become an appropriate abstraction for a broad class of large-scale MAS made of autonomous reactive agents. We argue that thus generalized CA can capture many important MAS properties at the level of agent ensembles and their long-term global behavior patterns.*

In this paper, we focus on the issue of inter-agent communication in CA. We propose sequential CA *as the first step towards the* genuinely asynchronous *CA as the ultimate CA-based abstraction for MAS insofar as the model of inter-agent communication is concerned. We then compare and contrast certain configuration space properties of* simple threshold sequential *CA with the corresponding properties of* parallel, perfectly synchronous *simple threshold CA. Specifically, we show that the possibility of "looping" in parallel threshold CA is* solely *due to the unrealistic assumption of perfect inter-agent communication synchrony.*

1 Introduction and Motivation

Multi-Agent Systems (MAS) are a research area where (distributed) artificial intelligence and distributed computing overlap [23]. Hence, research in MAS 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.* Among many abstract mathematical models of discrete dynamical systems, the one class that we find particularly simple yet useful for addressing many fundamental issues in distributed computing in general, and in large-scale multi-agent systems in particular, are the classical *cellular automata* [8, 18, 25, 26, 27] and some of their graph or network automata extensions and variants.

Cellular automata are discrete dynamical systems whose individual components are rather simple, yet that can exhibit highly complex and unpredictable behavior due to these simple components' mutual interaction and synergy. A CA is made of a finite or infinite regular 1-D, 2-D or higher-dimensional grid of nodes, where each node behaves like a finite state machine with a small number of states (typically, two or three). The nodes are interconnected together and can affect each other: the future state of a given node, in addition to its own current state, also depends on the current states of some of its near-by nodes. The "program" that tells a node to what value it should update its state, based on the states of these neighboring nodes, is deterministic and fixed; it is called the *local update rule.* All the CA nodes update (i) synchronously in parallel with each other, and (ii) according to the same update rule. The global dynamics of the entire CA is then obtained by composing together the effects of this local update rule on each individual node[1].

What are the important properties of the large-scale distributed computational and communication systems in general, and MAS in particular, that can be adequately captured by the CA-like models? Let's consider CA from distributed computing and multi-agent system perspectives. 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 uniform (i.e., *homogeneous*) across the entire system. In particular, CA and other related models capture the important distributed and multi-agent systems' properties of *locality* of interaction among the agents, and of the bounded speed of information and impact propagation.

Several modifications of the basic CA model along different dimensions can be argued to provide appropriate abstractions for the large-scale distributed computing and, in particular, for the *large-scale MAS made of reactive, autonomously executing agents.* We identify the following four as the most important:

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

[1] We will formally define all the necessary CA-related concepts in Section 2.

- *model of inter-agent communication* insofar as whether the agents locally compute synchronously or asynchronously, and whether they interact (i.e., 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 state of their environment is specified;

- *dynamic changes* of the MAS network topology, that would be captured by allowing the underlying cellular space of a cellular or graph automaton to change as a function of time.

Of the outlined four dimensions along which the classical CA can be readily generalized, this paper will focus on the second. Some implications of the perfect synchrony of the *parallel CA* node updates will be identified, and the CA behaviors studied once this physically unrealistic assumption is dropped. While the emphasis of this work will be on *Fair Sequential CA* and their properties, this paper can also be viewed as a prelude to studying *genuinely asynchronous CA* models as the appropriate and realistic abstraction for the large-scale multi-agent systems; see also discussion in *Section 4* of [21].

As already mentioned, the classical, parallel CA are characterized, among other properties, by *perfect synchrony* of the parallel node updates. This perfect synchrony is hard to justify on either physics or computer science grounds. By allowing the nodes to update one at a time, one arrives at a sequential version of *CA,* called *Sequential Cellular Automata (SCA),* and sequential versions of the corresponding more general graph automata. For instance, *Sequential Dynamical Systems (SDS)* are a class of sequential network automata that has been proposed as a mathematical model of computer simulations of distributed multi-agent infrastructures (see, e.g., [2, 3, 4, 5]).

However, these and other sequential cellular and network automata models, while more realistic than their synchronous parallel counterparts, still fall short of being an appropriate abstract model for distributed computation. Namely, a global clock, and therefore communication synchrony, is still assumed in all of these models. That is, the local computations of agents are 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 network 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 abstract models of behavior of biological systems capable of self-reproduction [18]. Subsequently, CA have been extensively studied in a great variety of application domains, but mostly in the context of physical sciences and, more specifically, of complex physical or biological systems and their dynamics (e.g., [9, 10, 25, 26]). However, CA can also be viewed as an abstraction of massively parallel computers (e.g, [8]).

We study in this paper a particular simple yet nontrivial class of CA from a distributed computing perspective. This class are the *threshold cellular automata.* We pose (and partially answer) some fundamental questions regarding the nature of the CA parallelism in the context of *(simple) threshold CA* [21, 4].

CA are an abstract computational model of *fine-grain parallelism* [8], in that the elementary operations executed at each node are rather simple and hence comparable to the basic operations performed by the computer hardware. In a classical CA, all the nodes execute their operations *logically simultaneously:* in general, the state of node x_i at time step $t+1$ is some simple function of the states of node x_i itself, and a set of its pre-specified neighbors at time t.

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 found in the literature, e.g., in [13]. It is important to understand, however, that even in case of what is referred to as *asynchronous cellular automata (ACA)* in the literature, the term *asynchrony* there applies to local updates (i.e., computations) *only,* but not to communication, since a tacit assumption of the globally accessible global clock still holds. Hence, we shall refer to this kind of (weakly asynchronous) CA as *sequential cellular automata (SCA).*

Definition 2.1. *A Cellular Space,* Γ, *is an ordered pair* (G, Q), *where*

- G *is a regular 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 quiescent state, denoted by* 0.

We denote the set of integer labels of the nodes (vertices) in Γ by L.

Definition 2.2. *A Cellular Automaton* **A** *is an ordered triple* (Γ, N, M), *where*

- Γ *is the cellular space;*

- N *is the fundamental neighborhood; and*

- M *is a* finite state machine *such that the input alphabet of* M *is* $Q^{|N|}$, *and the local transition function for each node is of the form* $\delta : Q^{|N|+1} \to Q$ *for CA with memory.*

The fundamental neighborhood N specifies what near-by nodes provide inputs to the update rule of a given node. In the classical *CA,* Γ is a regular graph that locally "looks the same everywhere"; in particular, the local neighborhood N is the same for each node in Γ.

The local transition rule δ specifies how each node updates its state, based on its current state, and the current states of its neighbors in N. By composing together the application of the local transition rule to each of the *CA's* nodes, we obtain *the global map* on the set of (global) configurations of a cellular automaton.

Definition 2.3. *A* Sequential Cellular Automaton (SCA) **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 all of whose elements are drawn from the set* L *of integers used in labelling 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 synchronous parallel *CA* with their sequential counterparts, rather than making a comparison between a given *CA* with a *particular SCA,* we compare the parallel *CA* computations with the computations of the corresponding *SCA* for *all* possible sequences of node updates. For that purpose, the following class of sequential automata is introduced:

Definition 2.4. *A Nondeterministic Interleavings Cellular Automaton (NICA)* ***I*** *is defined to be the union of all sequential automata* **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 the same as before, and the union is taken over* all *(finite and infinite) sequences* $s : \{1, 2, 3, ...\} \to L$**,** *where* L *is the set of integer labels of the nodes in* Γ**.**

We now introduce some terminology from physics that we find useful for characterizing *all possible computations* of a parallel or a sequential cellular automaton. To this end, a *(discrete) dynamical system* view of *CA* is helpful. A *phase space* (or *configuration space*) of a dynamical system is a directed graph where the vertices are the *global configurations* of the system, and directed edges correspond to the possible direct transitions from one global state to another.

As for any other kind of dynamical systems, we can define the fundamental, qualitatively distinct types of global configurations of a cellular automaton. We first define these types of dynamical system configurations for the *parallel CA*, and then briefly discuss how these definitions need to be modified in case of *SCA* and *NICA*. The classification 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 possible types of global 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* dynamic evolution reaches this configuration, it stays there "for a while" (i.e., for some finite number of sequential node update steps), 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 (see [21]).

3 On Temporal Cycles in Parallel and Sequential Threshold CA

We shall now compare and contrast the classical, perfectly synchronous *parallel CA* with their sequential counterparts, *SCA* and *NICA*, in the context of the *symmetric linear threshold functions.*

First, we define *(simple) linear threshold functions*, and the corresponding types of *(S)CA*.

Definition 3.1. *A Boolean-valued linear threshold function of* m *inputs,* $x_1, ..., x_m$*, is any function of the form*

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

where θ *is an appropriate threshold constant, and* $w_1, ..., w_m$ *are arbitrary (but fixed) real numbers*[2] *called weights.*

Definition 3.2. *A threshold automaton (threshold (S)CA) is a (parallel or sequential) 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 *Definition 3.1* with $\theta = k$ and an appropriate choice of weights w_i, $i = 1, ..., m$. We will consider *monotonically nondecreasing* Boolean threshold functions only; this, in particular, implies that the weights w_i are always nonnegative. We will also additionally assume δ to be a *symmetric function* of all of its inputs. That is, the *(S)CA* that we analyze have *symmetric, monotone Boolean functions* for their local update rules. We call such rules *simple threshold functions* [4, 21], and the *(S)CA* with simple threshold node update rules we refer to as *simple threshold (S)CA.*

[2] 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 nonnegative - that is, only *excitatory effects* of a node on its neighbors are allowed; see, e.g., [3, 4, 24, 25].

Definition 3.3. *A* simple *threshold* (S)CA *is a (sequential) cellular automaton whose local update rule δ is a monotone symmetric Boolean (threshold) function.*

In particular, if all the weights w_i are positive and equal to one another, then, without loss of generality, we may set them all equal to 1. This normalization of the weights w_j may also require an appropriate adjustment of the threshold value θ. The 1-D simple threshold *(S)CA* studied in the sequel will therefore have the node update functions of the general form

$$\delta(x_{i-r}, ..., \mathbf{x_i}, ..., x_{i+r}) = \begin{cases} 1, & \text{if } \sum_{j=-r}^{r} x_{i+j} \geq k \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where k is a fixed integer from the range $\{0, 1, ..., 2r + 1, 2r + 2\}$. For example, if the automaton rule radius is $r = 2$, and if $k = 2$, then the k-threshold *(S)CA* on a specified number of nodes in this case is just the 1-D (S)CA with the node update rule $\delta =$ *"at least 2 out of 5"*, meaning that the update rule evaluates to 1 if and only if at least two out of five of its inputs are currently equal to 1.

Beside the Boolean AND and OR functions, the most extensively studied simple threshold update rule is the $MAJORITY$ function, $\delta = MAJ$. For any radius $r \geq 1$ of a parallel or sequential 1-D CA with memory, if the automaton's update rule is MAJ, then any given node will update its state to 1 if and only if at least $r + 1$ of its $2r + 1$ inputs are equal to 1.

Due to the nature of the node update rules, cyclic behavior intuitively should not be expected in simple threshold cellular automata. This is, indeed, (almost) the case, as will be shown below. We attribute the importance of the results in this section to the following three factors:

- the local update rules are the simplest possible *nonadditive totalistic rules* (see, e.g., [25, 26]);

- given the rules, the cycles are not to be expected - yet they exist, and in the case of the synchronous parallel *CA only*; and, related to that observation,

- it is, for this class of automata, the parallel *CA* that provably have the more diverse possible dynamics: while there are no qualitative properties among the possible sequential computations that are not also present in the parallel case, the parallel threshold *CA* do exhibit a particular qualitative behavior that cannot be reproduced by any threshold *SCA*.

The results below hold for two-way infinite *1-D (S)CA,* as well as for finite *(S)CA* with the circular boundary conditions.

Theorem 3.1. *Parallel CA with $\delta = MAJ$ and $r = 1$ have temporal two-cycles. In contrast, for any* simple threshold Boolean 1-D Sequential CA $\mathbf{A}$ *with $r = 1$, and any sequence s of the node updates, the configuration space* **PS(A)** *of the sequential cellular automaton* $\mathbf{A}$ *is cycle-free.*

It turns out that the two-cycles in the *PS* of parallel CA with $\delta = MAJ$ are actually the only type of (proper) temporal cycles such CA can have. Indeed, for any *symmetric linear threshold update rule* δ, and any *finite* regular Cayley graph as the underlying cellular space, the following general result holds [8, 9]:

Proposition 3.1. *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 $C \in \mathbf{PS}(\mathbf{A})$, there exists a finite time step $t \geq 0$ such that $F^{t+2}(C) = F^t(C)$.*

In particular, this result implies that, in case of any *finite* symmetric threshold automaton, for any starting configuration C_0, there are *only two possible kinds of orbits:* upon repeated iteration, 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 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. The result 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 notion of a single computational step of an *SCA* is that of a single node updating its state. Thus, a single parallel step of a classical *CA* defined on an infinite underlying cellular space Γ is equivalent to 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* (esp. when the underlying Γ is infinite), and, more generally, in order 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 towards 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 Dynamical Systems (SDSs);* see, e.g., [3, 4, 5, 6]. In the infinite *SCA* case, on the other hand, the issue of fairness is nontrivial, and some form of *dove-tailing* of the sequential node updates may need to be imposed. In the sequel, we shall require from the sequences s of node updates of the *SCA* and *NICA* threshold automata to be fair in the sense defined below, without imposing any further restrictions or investigating how are such fair sequences of node updates to be generated in an actual distributed computing environment. For our present purposes, the following simple notion of fairness will suffice:

Definition 3.4. *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 $s(1) = s_1, s(2) = s_2, s(3) = s_3, ...$*

Let $s : N \to L$ be an arbitrary infinite sequence of elements from some domain L. Let $s^{[q]}$ denote the q-tail of s, i.e., $s^{[q]} = s_{q+1}, s_{q+2}, s_{q+3}, ...$

We state the following alternative characterizations of fair sequences:

Observation 3.1. *Let an infinite sequence $s : N \to L$ be given, where the set L is countable. Then the following four properties are equivalent to one another:*

(i) s *is* fair;
(ii) $\forall n \in N$, $s^{[n]}$ *is* fair;
(iii) $(\forall x \in L)(\forall n \in N)(\exists n' > n)(s(n') = x)$
(iv) $\forall n \in N$, $s^{[n]} : \{n+1, n+2, ...\} \to L$ *is* onto L.

Now that we have adopted an appropriate notion of *fairness,* we can establish the following generalization of *Proposition 3.1* applicable to both finite and infinite *1-D (S)CA.*

Proposition 3.2. *Let a parallel* CA *or a sequential* SCA $\mathbf{A}$ *be defined over a finite or infinite 1-D cellular space, with a finite rule radius $r \geq 1$. Let this cellular automaton's local update rule be a simple threshold function. Let's also assume, in the sequential cases, that the fairness condition from* Definition 3.4 *holds. Then for any starting configuration $C_0 \in \mathbf{PS}(\mathbf{A})$ whatsoever, and any* finite subconfiguration $C \subseteq C_0$, *there exists a time step $t \geq 0$ such that*

$$F^{t+2}(C) = F^t(C) \tag{3}$$

where, in the case of fair SCA, *the* Eqn. (3) *can be replaced with*

$$F^{t+1}(C) = F^t(C) \tag{4}$$

Proposition 3.3. *Let the assumptions from* Proposition 3.2 *hold, and let the underlying threshold rule be $\delta = MAJ$. Then for all configurations $C \in \mathbf{PS}(\mathbf{A})$ whatsoever in the finite cases, and for all configurations $C \in \mathbf{PS}(\mathbf{A})$ such that C has a* finite support *when $\mathbf{\Gamma}(\mathbf{A})$ 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 3.2-3.3,* and the dynamic evolution of the full such global configurations is monitored, then the only additional possibility is that the particular *(S)CA* computation fails to finitely converge altogether. In that case, and under the fairness assumption in the case of *SCA,* the limiting configuration $lim_{t\to\infty} F^t(C) = C^{lim}$ can be shown to be a (stable) fixed point.

That is, if the computation of a SCA starting from some configuration C converges at all, it actually has to converge to a fixed point.

Theorem 3.2. *The following dichotomy holds:*

(i) All 1-D (parallel) CA *with any odd $r \geq 1$, the local rule $\delta =$* 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 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, linear 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 nontrivial) temporal cycles. In contrast, the corresponding classes of *SCA* (and therefore *NICA*) can never cycle. We also observe that, given any sequence of node updates of a finite threshold *SCA*, if this sequence satisfies an appropriate *fairness condition,* then the computation of such a threshold SCA $\mathbf{A}$ is guaranteed to converge to a fixed-point (sub)configuration on any finite subset of the nodes in $\mathbf{\Gamma}(\mathbf{A})$.

The cycle-freeness of the threshold *SCA* and *NICA* holds irrespective of the choice of a sequential update ordering (and, extending to infinite *SCA,* temporal cycles cannot be obtained even "in the limit"[4]). Hence, the existence of temporal cycles in the parallel cases is entirely due to the assumption of *perfect synchrony* of the parallel node updates.

The practical implication of this result in the context of distributed computing and multi-agent systems is as follows. If an actual distributed computing infrastructure can be approximated as a ring network of communicating finite state machines (FSMs), where each FSM behaves according to a linear threshold rule, it can be safely concluded that, as long as each component functions properly and provided that there exists a fixed finite bound on the ratio of computational speeds of different components, if allowed to execute autonomously, such an infrastructure will always eventually converge to a steady state.

4 Summary

We have presented some early steps in modeling the large-scale MAS and their global properties by the cellular automata based complex system models. In order to make these models a suitable abstraction for distributed computing in general, and MAS in particular, several properties of the classical CA need to be modified.

[3]There are also *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 is required; we leave this discussion out, however, for the sake of clarity and space constraints.

[4]That is, via infinitely long computations, obtained by allowing arbitrary infinite sequences of individual node updates.

In this short paper, we have focused on the CA communication models and their implications. In particular, we have studied a class of simple totalistic cellular automata [25, 26] when the unrealistic assumptions of *perfect synchrony* and *instantaneous unbounded parallelism* are dropped [21]. We have compared and contrasted sequential vs. parallel simple threshold CA, and showed that the perfectly synchronous parallel CA can exhibit qualitative behaviors that cannot be reproduced by any of the corresponding sequential CA, irrespective of the choice of a node update sequence.

It is our hope that the further study of cellular automata and their variants and extensions, such as the *SCA, NICA, SDS* and *Asynchronous CA* models proposed or referred to herewith, as well as other, similar in spirit network automata models, will prove very useful in the context of high-level mathematical modeling and analysis of many important problems in distributed computing and distributed AI.

Acknowledgements: The author thanks Gul Agha, Michael Loui and Mahesh Viswanathan (all at University of Illinois), as well as the anonymous referees, for feedback and useful suggestions. This work was supported in part by the ONR MURI Grant awarded to Gul Agha, contract number N00014-02-1-0715.

References

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

[2] C. Barrett and C. Reidys, "Elements of a theory of computer simulation I: sequential CA over random graphs", *Applied Math. and Computation,* vol. 98 (2-3), 1999

[3] C. Barrett, H. Hunt, M. Marathe, S. S. Ravi, D. Rosenkrantz, R. Stearns, and P. Tosic, "Gardens of Eden and Fixed Points in Sequential Dynamical Systems", *Discrete Math. and Theoretical Comp. Sci.,* Proc. AA (DM-CCG), July 2001

[4] C. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns, "Reachability problems for sequential dynamical systems with threshold functions", *Theoretical Computer Sci.,* vol. 295, issues 1-3, pp. 41-64, Feb. 2003

[5] C. Barrett, H. Mortveit, and C. Reidys, "Elements of a theory of computer simulation II: sequential dynamical systems", *Applied Math. and Computation,* vol. 107 (2-3), 2000

[6] C. Barrett, H. Mortveit, and C. Reidys, "Elements of a theory of computer simulation III: equivalence of sequential dynamical systems", *Applied Math. and Computation,* vol. 122 (3), 2001

[7] I. Czaja, R. J. van Glabbek, U. Goltz, "Interleaving semantics and action refinement with atomic choice", in *"Advances in Petri Nets"* (G. Rozenberg, ed.), LNCS 609, Springer-Verlag, 1992

[8] Max Garzon, *"Models of Massive Parallelism: Analysis of Cellular Automata and Neural Networks",* Springer, 1995

[9] E. Goles, S. Martinez, *"Neural and Automata Networks: Dynamical behavior and Applications",* Math. and Its Applications series (vol. 58), Kluwer, 1990

[10] E. Goles, S. Martinez (eds.), *"Cellular Automata and Complex Systems",* Nonlinear Phenomena and Complex Systems series, Kluwer, 1999

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

[12] C. A. R. Hoare, *"Communicating Sequential Processes",* Prentice Hall, 1985

[13] T. E. Ingerson and R. L. Buvel, "Structure in asynchronous cellular automata", *Physica D: Nonlinear Phenomena,* Vol. 10, Issues 1-2, January 1984

[14] S. A. Kauffman, "Emergent properties in random complex automata" (ibid.)

[15] Robin Milner, *"A Calculus of Communicating Systems",* Springer-Verlag Lecture Notes in Computer Science (LNCS), 1980

[16] Robin Milner, "Calculi for synchrony and asynchrony", Theoretical Computer Sci. 25, pp. 267-310, 1983

[17] Robin Milner, *"Communication and Concurrency",* Prentice-Hall, 1989

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

[19] C. Reidys, "On acyclic orientations and sequential dynamical systems", *Adv. Appl. Math.,* vol. 27, 2001

[20] P. Tosic, "A Perspective on the Future of Massively Parallel Computing: Fine-Grain vs. Coarse-Grain Parallel Models", *Proc. ACM Computing Frontiers '04,* Ischia, Italy, April 2004

[21] P. Tosic, G. Agha, "Concurrency vs. Sequential Interleavings in 1-D Threshold Cellular Automata", APDCM Workshop within IPDPS'04, Santa Fe, New Mexico, USA, April 2004 (in *Proc. IEEE-IPDPS 2004*)

[22] P. Tosic, G. Agha, "Characterizing Configuration Spaces of Simple Threshold Cellular Automata", *Proc. ACRI'04,* Amsterdam, The Netherlands, Oct. 25-28, 2004 (in Springer-Verlag LNCS series, vol. 3305, pp. 861 - 870)

[23] G. Weiss (ed.), *"Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence",* The MIT Press, Cambridge, Massachusetts, 1999

[24] S. Wolfram "Twenty problems in the theory of CA", *Physica Scripta,* vol. 9, 1985

[25] S. Wolfram (ed.), *"Theory and applications of CA",* World Scientific, Singapore, 1986

[26] Stephen Wolfram, *"Cellular Automata and Complexity (collected papers)",* Addison-Wesley, 1994

[27] Stephen Wolfram, *"A New Kind of Science",* Wolfram Media, Inc., 2002