Characterizing Configuration Spaces of Simple Threshold Cellular Automata

Predrag T. Tosic and Gul A. Agha

Open Systems Laboratory, Department of Computer Science
University of Illinois at Urbana-Champaign
Mailing address: Siebel Center for Computer Science,
201 N. Goodwin Ave., Urbana, IL 61801, USA
p-tosic@cs.uiuc.edu, agha@cs.uiuc.edu

Abstract. We study herewith *the simple threshold cellular automata (CA),* as perhaps the simplest broad class of *CA* with non-additive (i.e., non-linear and non-affine) local update rules. We characterize all possible computations of the most interesting rule for such *CA,* namely, the *Majority (MAJ)* rule, both in the classical, parallel CA case, and in case of the corresponding sequential CA where the nodes update sequentially, one at a time. We compare and contrast the configuration spaces of arbitrary simple threshold automata in those two cases, and point out that some parallel threshold CA cannot be simulated by any of their sequential counterparts. We show that the temporal cycles exist only in case of (some) parallel simple threshold CA, but can never take place in sequential threshold CA. We also show that most threshold *CA* have very few fixed point configurations and few (if any) cycle configurations, and that, while the *MAJ* sequential and parallel *CA* may have many fixed points, nonetheless "almost all" configurations, in both parallel and sequential cases, are transient states.

1 Introduction and Motivation

Cellular automata (CA) were originally introduced as an abstract mathematical model of the behavior of biological systems capable of self-reproduction [15]. Subsequently, variants of *CA* have been extensively studied in a great variety of application domains, predominantly in the context of complex physical or biological systems and their dynamics (e.g., [20, 21, 22]). However, *CA* can also be viewed as an abstraction of massively parallel computers (e.g, [7]). Herein, we study a particular simple yet nontrivial class of *CA* from a computer science perspective. This class are the *threshold cellular automata.* In the context of such *CA,* we shall first compare and contrast the configuration spaces of the classical, concurrent *CA* and their sequential analogues. We will then pick a particular threshold node update rule, and fully characterize possible computations in both parallel and sequential cases for the one-dimensional automata.

Cellular automata CA are an abstract computational model of *fine-grain parallelism* [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. In a classical, that is, concurrently executing *CA,* whether finite or infinite, all the nodes execute their operations *logically simultaneously:* the state of a node x_i at time step

$t + 1$ is some simple function of the states (i) of the node x_i itself, and (ii) of a set of its pre-specified neighbors, at time t.

We consider herewith the sequential version of *CA*, heretofore abridged to *SCA*, and compare such sequential *CA* with the classical, *parallel* (*concurrent*) *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.

We also fully characterize the possible computations of the most interesting case of *threshold cellular automata*, namely, the *(S)CA* with the *Majority* node update rule.

An important remark is that we use the terms *parallel* and *concurrent* as synonyms throughout the paper. This is perhaps not the most standard convention, but we are not alone in not making the distinction between the two terms (cf. discussion in [16]). Moreover, by a *parallel* (equivalently, *concurrent*) *computation* we shall mean actions of several processing units that are carried out *logically* (if not necessarily *physically*) *simultaneously*. In particular, when referring to *parallel* or *concurrent* computation, we do assume a *perfect synchrony.*

2 Cellular Automata and Types of Their Configurations

We follow [7] and define classical (that is, synchronous and concurrent) *CA* in two steps: by first defining the notion of a *cellular space,* and subsequently that of a *cellular automaton* defined over an appropriate cellular space.

Definition 1: A *Cellular Space,* Γ, is an ordered pair (G, Q) where G is a regular graph (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 (vertices) in Γ by L.

Definition 2: A *Cellular Automaton (CA),* **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* such that the input alphabet of M is $Q^{|N|}$, and the local transition function (update rule) for each node is of the form $\delta : Q^{|N|+1} \rightarrow Q$ for *CA with memory*, and $\delta : Q^{|N|} \rightarrow Q$ for *memoryless CA.*

Some of our results pertain to a comparison and contrast between the classical, concurrent threshold CA and their sequential counterparts, the threshold SCA.

Definition 3: A *Sequential Cellular Automaton (SCA)* **S** is an ordered quadruple (Γ, N, M, s), where Γ, N and M are as in *Def. 2,* and s is a 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 threshold 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. To that end, the following convenient terminology is introduced:

Definition 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, $\mathbf{I} = \cup_s (\Gamma, N, M, s)$, 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).

Since our goal is to characterize *all* possible computations of parallel and sequential threshold *CA*, a *(discrete) dynamical system* view of *CA* will be useful. A *phase space* of a dynamical system is a (finite or infinite, as appropriate) directed graph where the vertices are the *global configurations* (or *global states*) of the system, and directed edges correspond to possible transitions from one global state to another. We now define the fundamental, qualitatively distinct types of (global) configurations that a classical (parallel) cellular automaton can find itself in.

Definition 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 *(proper) cycle configuration (CC)* is a state that, if once reached, will be revisited infinitely often with a fixed, finite period of 2 or greater. A *transient configuration (TC)* is a state that, once reached, is never going to be revisited again.

In particular, FPs are a special, degenerate case of recurrent states whose period is 1. Due to their deterministic evolution, any configuration of a classical, parallel *CA* belongs to exactly one of these basic configuration types, i.e., it is a FP, a proper CC, or a TC. On the other hand, if one considers *sequential CA* so that *arbitrary* node update orderings are permitted, that is, if one considers *NICA* automata, 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 a *SCA* configuration such that, once the dynamics 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 a *SCA* can be simultaneously a (quasi-)FP and a (quasi-)CC (see, e.g., the example in [19]). For simplicity, heretofore we shall refer to a configuration x of a *NICA* as a *pseudo fixed point* if there exists some infinite sequence of node updates s such that x is a FP in the usual sense when the corresponding *SCA's* nodes update according to the ordering s. A global configuration of a *NICA* is a *proper* FP iff it is a fixed point of each corresponding *SCA*, that is, for every sequence of node updates s. Similarly, we consider a global configuration y of a *NICA* to be a cycle state, if there exists an infinite sequence of the node updates s' such that, if the corresponding *SCA's* nodes update according to s', then y is a recurrent state and, moreover, y is not a *proper FP*. Thus, in general, a global configuration of a *NICA* automaton can be simultaneously a (pseudo) FP, a CC and a TC (with respect to different node update sequences s)[1].

[1] When the allowable sequences of node updates $s : \{1, 2, 3, ...\} \to L$ are required to be infinite and *fair* so that, in particular, every (infinite) tail $s^{[n]} : \{n, n+1, n+2, ...\} \to L$ is *onto L*, then *pseudo fixed points* and *proper fixed points* in *NICA* can be shown to coincide with one another and, moreover, with the "ordinary" *FPs* for parallel *CA*. For the special case when L is finite and s is required to be an *ad infinitum* repeated permutation see, e.g., [3, 4].

Definition 6: A *1-D cellular automaton of radius* r ($r \geq 1$) is a *CA* defined over a one-dimensional string of nodes, such that each node's next state depends on the current states of its neighbors to the left and to the right that are no more than r nodes away (and, in case of the *CA with memory,* on the current state of that node itself).

We adopt the following conventions and terminology. Throughout, only *Boolean CA* and *SCA/NICA* are considered; in particular, the set of possible states of any node is $\{0, 1\}$. The terms "monotone symmetric" and "symmetric (linear) threshold" functions/update rules/automata are used interchangeably. Similarly, the terms "(global) dynamics" and "(global) computation" are used synonymously. Also, unless explicitly stated otherwise, automata *with memory* are assumed. The default *infinite* cellular space Γ is a two-way infinite line. The default *finite* Γ is a ring with an appropriate number of nodes[2]. The terms "phase space" and "configuration space" will be used synonymously, as well, and sometimes abridged to *PS*.

3 Properties of 1-D Simple Boolean Threshold CA and SCA

Herein, we compare and contrast the classical, parallel *CA* with their sequential counterparts, *SCA* and *NICA,* in the context of the simplest (nonlinear) local update rules possible, namely, the *Boolean linear threshold rules.* Moreover, we choose these threshold functions to be *symmetric,* so that the resulting *CA* are also *totalistic* (see, e.g., [7] or [21]). We show the fundamental difference in the configuration spaces, and therefore possible computations, in case of the classical, concurrent threshold automata on one, and the sequential threshold cellular automata, on the other hand: while the former can have temporal cycles (of length two), the computations of the latter either do not converge at all after any finite number of sequential steps, or, if the convergence does take place, it is *necessarily* to a fixed point.

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

Definition 7: A *Boolean-valued linear threshold function* of n inputs, $x_1, ..., x_n$, is any function of the form

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

where θ is an appropriate *threshold constant,* and w_i are real-valued *weights.*

Definition 8: A *threshold cellular automaton* is a (parallel or sequential) cellular automaton such that its node update rule δ is a *Boolean-valued linear threshold function.*

[2] It turns out, that circular boundary conditions are important for some of our technical results. Likewise, some results about the phase space properties of concurrent and sequential threshold *CA* may require (i) a certain minimal number of nodes and (ii) that the number of nodes be, e.g., even, divisible by four, or the like. Heretofore, we shall assume a sufficient number of nodes that "works" in the particular situation, without detailed elaborations.

Definition 9: A *simple* threshold *(S)CA* is an automaton whose local update rule δ is a monotone symmetric Boolean (threshold) function.

Throughout, whenever we say *a threshold automaton (threshold CA)*, we shall mean *simple threshold automaton (threshold CA)* - unless explicitly stated otherwise.

Due to the nature of the node update rules, cyclic behavior intuitively should not be expected in these simple threshold automata. This is, generally, (almost) the case, as will be shown below. We argue that the importance of the results in this section largely stems from the following three factors: (i) the local update rules are the simplest nonlinear totalistic rules one can think of; (ii) given the rules, the cycles are not to be expected - yet they exist, and in the case of classical, parallel *CA only;* and, related to that observation, (iii) it is, for this class of *(S)CA*, the parallel *CA* that exhibit the more interesting behavior than any corresponding sequential *SCA* (and consequently also *NICA*) [19], and, in particular, while there is nothing (qualitatively) among the possible sequential computations that is not present in the parallel case, the classical parallel threshold *CA* are capable of a particular qualitative behavior - namely, they may have nontrivial temporal cycles - that cannot be reproduced by *any* simple threshold *SCA* (and, therefore, also threshold *NICA*).

The results below hold for the two-way infinite *1-D CA,* as well as for the finite *CA* and *SCA* with sufficiently many nodes and circular boundary conditions.

Lemma 1: (i) A 1-D classical (i.e., parallel) *CA* with $r = 1$ and the *Majority* update rule has (finite) temporal cycles in the phase space (*PS*). In contrast, (ii) *1-D Sequential CA* with $r = 1$ and the *Majority* update rule do not have any (finite) cycles in the phase space, *irrespective* of the sequential node update order s. $\diamond$

Remarks: In case of infinite sequential *SCA* as in the *Lemma* above, a nontrivial cycle configuration does not exist even in the limit. In finite cases, s is an arbitrary sequence of an *SCA* nodes' indices, not necessarily a (repeated) permutation.

We thus conclude that *NICA* with $\delta = MAJ$ and $r = 1$ are temporal cycle-free. Moreover, it turns out that, even if we consider local update rules δ other than the *MAJ* rule, yet restrict δ to *monotone symmetric Boolean functions,* such sequential *CA* still do not have any temporal cycles.

Lemma 2: For any *Monotone Symmetric Boolean 1-D Sequential CA* **S** with $r = 1$, and any sequential update order s, the phase space *PS(***S***)* is cycle-free. $\diamond$

Similar results to those in *Lemmata 1-2* also hold for *1-D CA* with radius $r \geq 2$.

Theorem 1: (i) *1-D (parallel) CA* with $r \geq 1$ and with the *Majority* node update rule have (finite) cycles in the phase space. (ii) Any *1-D SCA* with $\delta = MAJ$ or any other monotone symmetric Boolean node update rule, $r \geq 1$ and any sequential order s of the node updates has a cycle-free phase space. $\diamond$

Remarks: The claims of *Thm. 1* hold both for the finite *(S)CA* (provided that they have sufficiently many nodes, an even number of nodes in case of the *CA* with cycles, and assuming the circular boundary conditions in *part (i)*), and for the infinite *(S)CA.* We also observe that several variants of the result in *Theorem 1 (ii)* can be found in the literature. When the sequence of node updates of a finite *SCA* is periodic, with a single period a fixed permutation of the nodes, the temporal cycle-freeness of sequential *CA* and many other properties can be found in [8] and references therein. In [4], fixed permutation of the sequential node updates is also required, but the underlying cellular space Γ is allowed to be an arbitrary finite graph, and different nodes are allowed to compute *different* simple k-threshold functions.

As an immediate consequence of the results presented thus far, we have

Corollary 1: For all $r \geq 1$, there exists a *monotone symmetric CA* (that is, a *threshold automaton*) $\mathbf{A}$ such that $\mathbf{A}$ has finite temporal cycles in the phase space.

Some of the results for *(S)CA* with $\delta = MAJ$ do extend to some, but by no means all, other simple threshold *(S)CA* defined over the same cellular spaces. For instance, consider the k-threshold functions with $r = 2$. There are five nontrivial such functions, for $k \in \{1, 2, 3, 4, 5\}$. The 1-threshold function is Boolean *OR* function (in this case, on $2r + 1 = 5$ inputs), and the corresponding *CA* do not have temporal cycles; likewise with the "5-threshold" *CA,* that update according to Boolean *AND* on five inputs. However, in addition to *Majority* (i.e., 3-threshold), it is easy to show that 2-threshold (and therefore, by symmetry, also 4-threshold) such *CA* with $r = 2$ do have temporal two-cycles; for example, in the 2-threshold case, for *CA* defined over an infinite line, $\{(1000)^\omega, (0010)^\omega\}$ is a two-cycle.

We now relate our results thus far to what has been already known about simple threshold *CA* and their phase space properties. In particular, the only recurrent types of configurations we have identified thus far are FPs (in the sequential case), and FPs and two-cycles, in the concurrent *CA* case. This is not a coincidence.

It turns out that the two-cycles in the *PS* of the parallel *CA* with $\delta = MAJ$ are actually the only type of (proper) temporal cycles such cellular automata 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 (see [7, 8]):

Proposition 1: Let a classical *CA* $\mathbf{A} = (\Gamma, N, T)$ be such that Γ is finite and the underlying local rule of T is an elementary symmetric threshold function. Then for all configurations $C \in PS(\mathbf{A})$, there exists $t \geq 0$ such that $T^{t+2}(C) = T^t(C)$. $\diamond$

In particular, this result implies that, in case of any finite simple threshold automaton, and for any starting configuration C_0, there are only two possible kinds of orbits: upon repeated iteration, after finitely many steps, the computation either converges to a fixed point configuration, or else it converges to a two-cycle[3].

We now specifically focus on $\delta = MAJ$ 1-D *CA,* with an emphasis on the infinite case, and completely characterize the configuration spaces of such threshold automata. In particular, in the $\Gamma =$ *infinite line* case, we show that the cycle configurations are rather rare, that fixed point configurations are quite numerous - yet still relatively rare in a sense to be discussed below, and that *almost all* configurations of these threshold automata are transient states.

Heretofore, insofar as the *SCA* and *NICA* automata were concerned, for the most part we have allowed entirely *arbitrary* sequences s of node updates, or at least *arbitrary infinite* such sequences. In order to carry the results on FPs and TCs of (parallel) *MAJ CA* over to the sequential automata with $\delta = MAJ$ (and, when applicable, other

[3] If one considers *threshold (S)CA* defined over infinite Γ, the only additional possibility is that such automaton's dynamic evolution fails to converge after any finite number of steps.

simple threshold rules) as well, throughout the rest of the paper we will allow *fair sequences only:* that is, we shall now consider only those threshold *SCA* (and *NICA*) where each node gets its turn to update *infinitely often.* In particular, this ensures that (i) any pseudo FP of a given *NICA* is also a proper FP, and (ii) the FPs of a given parallel *CA* coincide with the (proper) FPs of the corresponding *SCA* and *NICA.*

We begin with some simple observations about the nature of various configurations in the *(S)CA* with $\delta = MAJ$ and $r = 1$. We shall subsequently generalize most of these results to arbitrary $r \geq 1$. We first recall that, for such *(S)CA* with $r = 1$, two adjacent nodes of the same value are stable. That is, 11 and 00 are stable sub-configurations. Consider now the starting sub-configuration $x_{i-1}x_ix_{i+1}$ = 101. In the parallel case, at the next time step, $x_i \rightarrow 1$. Hence, no FP configuration of a parallel *CA* can contain 101 as a sub-configuration. In the sequential case, assuming fairness, x_i will eventually have to update. If, at that time, it is still the case that $x_{i-1} = x_{i+1} = 1$, then $x_i \rightarrow 1$, and $x_{i-1}x_ix_{i+1} \rightarrow 111$, which is stable. Else, at least one of x_{i-1}, x_{i+1} has already "flipped" into 0. Without loss of generality, let's assume $x_{i-1} = 0$. Then $x_{i-1}x_i$ = 00, which is stable; so, in particular, $x_{i-1}x_ix_{i+1}$ will never go back to the original 101. By symmetry of $\delta = MAJ$ with respect to 0 and 1, the same line of reasoning applies to the sub-configuration $x_{i-1}x_ix_{i+1}$ = 010. In particular, the following properties hold:

Lemma 3: A fixed point configuration of a *1D-(S)CA* with δ = *Majority* and $r = 1$ cannot contain sub-configurations 101 or 010. Similarly, a cycle configuration of such a *1D-(S)CA* cannot contain sub-configurations 00 or 11. $\diamond$

Of course, we have already known that, in the sequential case, no cycle states exist, period. In case of the parallel threshold *CA,* by virtue of determinism, a complete characterization of each of the three basic types of configurations (FPs, CCs, TCs) is now almost immediate:

Lemma 4: The FPs of the *1D-(S)CA* with $\delta = MAJ$ and $r = 1$ are precisely of the form $(000^*+111^*)^*$. The CCs of such *1D-CA* exist only in the concurrent case, and the temporal cycles are precisely of the form $\{(10)^*, (01)^*\}$. All other configurations are *transient states,* that is, TCs are precisely the configurations that contain both (i) 000^* or 111^* (or both), and (ii) 101 or 010 (or both) as their sub-configurations. In addition, the CCs in the parallel case become TCs in all corresponding sequential cases. $\diamond$

Some generalizations to arbitrary (finite) rule radii r are now immediate. For instance, given any such $r \geq 1$, the finite sub-configurations 0^{r+1} and 1^{r+1} are stable with respect to $\delta = MAJ$ update rule applied either in parallel or sequentially; consequently, any configuration of the form $(0^{r+1}0^* + 1^{r+1}1^*)^*$, for both finite and infinite *(S)CA,* is a fixed point. This characterization, only with a considerably different notation, has been known for the case of configurations with *compact support* for a relatively long time; see, e.g., *Chapter 4* in [8]. On the other hand, fully characterizing CCs (and, consequently, also TCs) in case of finite or infinite (parallel) *CA* is more complicated than in the simplest case with $r = 1$. For example, for $r \geq 1$ odd, and Γ = *infinite line,* $\{(10)^\omega, (01)^\omega\}$ is a two-cycle, whereas for $r \geq 2$ even, each of $(10)^\omega$, $(01)^\omega$ is a fixed point. However, for all $r \geq 1$, the corresponding (parallel) *CA* are guaranteed to have some temporal cycles, namely, given $r \geq 1$, the doubleton of states $\{(1^r0^r)^\omega, (0^r1^r)^\omega\}$ forms a temporal two-cycle.

Lemma 5: Given any (finite or infinite) threshold *(S)CA*, one of the following two properties always holds: either (i) this threshold automaton does not have proper cycles and cycle states; or (ii) if there are cycle states in the *PS* of this automaton, then none of those cycle states has any incoming transients. $\diamond$

Moreover, if there are any (two-)cycles, the number of these temporal cycles and therefore of the cycle states is, statistically speaking, negligible:

Lemma 6: Given an infinite *MAJ CA* and a finite radius of the node update rules $r \geq 1$, among uncountably many ($2^{\aleph_0}$, to be precise) global configurations of such a *CA*, there are only finitely many (proper) cycle states. $\diamond$

On the other hand, fixed points of some threshold automata are *much more numerous* than the CCs. The most striking are the *MAJ (S)CA* with their abundance of FPs. Namely, the cardinality of the set of FPs, in case of $\delta = MAJ$ and (countably) infinite cellular spaces, equals the cardinality of the entire *PS:*

Theorem 2: An infinite *1D-(S)CA* with $\delta = MAJ$ and any $r \geq 1$ has *uncountably many* fixed points. $\diamond$

The above result is another evidence that "not all threshold *(S)CA* are born equal". It suffices to consider only 1D, infinite *CA* to see a rather dramatic difference. Namely, in contrast to the $\delta = MAJ$ *CA,* the *CA* with memory and with $\delta \in \{OR, AND\}$ (i) do not have any temporal cycles, and (ii) have *exactly two* FPs, namely, 0^ω and 1^ω. Other threshold *CA* may have temporal cycles, as we have already shown, but they still have only a finite number of FPs.

We have just argued that *1-D infinite MAJ (S)CA* have uncountably many FPs. However, these FPs are, when compared to the transient states, still but a few. To see this, let's assume that a "random" global configuration is obtained by "picking" each site's value to be either 0 or 1 at random, with equal probability, and so that assigning a value to one site is independent of the value assignment to any of the other sites. Then the following result holds:

Lemma 7: If a global configuration of an infinite threshold automaton is selected "at random", that is, by assigning each node's value independently and according to a toss of a fair coin, then, with probability 1, this randomly chosen configuration will be a transient state. $\diamond$

Moreover, the "unbiased randomness", while sufficient, is certainly not necessary. In particular, assigning bit values according to outcomes of tossing a coin with a fixed bias also yields transient states being of probability one.

Theorem 3: Let p be any real number such that $0 < p < 1$, and let the probability of a site in a global configuration of a threshold automaton being in state 1 be equal to p (so that the probability of this site's state being 0 is equal to $q = 1 - p$). If a global configuration of this threshold automaton is selected "at random" where the state of each node is an *i.i.d. discrete random variable* according to the probability distribution specified by p, then, with probability 1, this global configuration will be a transient state. $\diamond$

In case of the finite threshold *(S)CA,* as the number of nodes, N, grows, the fraction of the total of 2^N global configurations that are TCs will also tend to grow.

In particular, under the same assumptions as above, in the limit, as $N \to \infty$, the probability that a randomly picked configuration, C, is a transient state approaches 1:

$$lim_{N\to\infty}\ Pr(C\ is\ transient)\ =\ 1 \tag{2}$$

Thus, a fairly complete characterization of the configuration spaces of threshold *CA/SCA/NICA* over finite and infinite 1-D cellular spaces can be given. In particular, under a simple and reasonable definition of what is meant by a "randomly chosen" global configuration in the infinite threshold *CA* case, *almost every* configuration of such a *CA* is a TC. However, when it comes to the number of fixed points, the striking contrast between $\delta = MAJ$ and all other threshold rules remains: in the infinite Γ cases, the *MAJ CA* have uncountably many FPs, whereas all other simple threshold *CA* have only finitely many FPs. The same characterizations hold for the *proper* FPs of the corresponding *simple threshold NICA* automata.

4 Conclusion

The theme of this work is a study of the fundamental configuration space properties of *simple threshold cellular automata,* both when the nodes update synchronously in parallel, and when they update sequentially, one at a time.

Motivated by the well-known notion of the sequential interleaving semantics of concurrency, we apply the *"interleaving semantics"* metaphor to the parallel *CA* and thus motivate the study of sequential cellular automata, *SCA* and *NICA,* and the comparison and contrast between *SCA* and *NICA* on one, and the classical, concurrent *CA,* on the other hand [19]. We have shown that even in this simplistic context, the perfect synchrony of the classical *CA* node updates has some important implications, and that the sequential *CA* cannot capture certain aspects of their parallel counterparts' behavior. Hence, simple as they may be, the basic operations (local node updates) in classical *CA* cannot always be considered atomic. Thus we find it reasonable to consider a single local node update to be made of an ordered sequence of finer elementary operations: (1) fetching ("receiving"?) all the neighbors' values, (ii) updating one's own state according to the update rule δ, and (iii) making available ("sending"?) one's new state to the neighbors.

We also study in some detail perhaps the most interesting of all simple threshold rules, namely, the *Majority* rule. In particular, we characterize all three fundamental types of configurations (transient states, cycle states and fixed point states) in case of finite and infinite *1D-CA* with $\delta = MAJ$ for various finite rule radii $r \geq 1$. We show that CCs are, indeed, a rare exception in such *MAJ CA,* and that, for instance, the infinite *MAJ (S)CA* have uncountably many FPs, in a huge contrast to other simple threshold rules that have only a handful of FPs. We also show that, assuming a random configuration is chosen via independently assigning to each node its state value by tossing a (not necessarily fair) coin, it is very likely, for a sufficiently large number of the automaton's nodes, that this randomly chosen configuration is a TC.

To summarize, the class of the simple threshold *CA, SCA,* and *NICA* is (i) relatively broad and interesting, and (ii) nonlinear (non-additive), yet (iii) all of these automata's long-term behavior patterns can be readily characterized and effectively predicted.

Acknowledgments: The work presented herein was supported by the *DARPA IPTO TASK Program*, contract number *F30602-00-2-0586*.

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. & Comput., 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. & 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", TCS 1-3: 41-64, 2003
5. C. Barrett, H. Mortveit, and C. Reidys, "Elements of a theory of computer simulation II: sequential dynamical systems", Applied Math. & Comput. 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", Appl. Math. & Comput. vol. 122(3), 2001
7. Max Garzon, "Models of Massive Parallelism: Analysis of Cellular Automata and Neural Networks", Springer, 1995
8. E. Goles, S. Martinez, "Neural and Automata Networks: Dynamical Behavior and Applications", Math. & Its Applications series (vol. 58), Kluwer, 1990
9. E. Goles, S. Martinez (eds.), "Cellular Automata and Complex Systems", Nonlinear Phenomena and Complex Systems series, Kluwer, 1999
10. T. E. Ingerson and R. L. Buvel, "Structure in asynchronous cellular automata", Physica D: Nonlinear Phenomena, vol. 10 (1-2), Jan. 1984
11. S. A. Kauffman, "Emergent properties in random complex automata", Physica D: Nonlinear Phenomena, vol. 10 (1-2), Jan. 1984
12. Robin Milner, "A Calculus of Communicating Systems", Lecture Notes Comp. Sci., Springer, Berlin, 1989
13. Robin Milner, "Calculi for synchrony and asynchrony", Theoretical Comp. Sci. 25, Elsevier, 1983
14. Robin Milner, "Communication and Concurrency", C. A. R. Hoare series ed., Prentice-Hall Int'l, 1989
15. John von Neumann, "Theory of Self-Reproducing Automata", edited and completed by A. W. Burks, Univ. of Illinois Press, Urbana, 1966
16. J. C. Reynolds, "Theories of Programming Languages", Cambridge Univ. Press, 1998
17. Ravi Sethi, "Programming Languages: Concepts & Constructs", 2nd ed., Addison-Wesley, 1996
18. K. Sutner, "Computation theory of cellular automata", MFCS98 Satellite Workshop on CA, Brno, Czech Rep., 1998
19. P. Tosic, G. Agha, "Concurrency vs. Sequential Interleavings in 1-D Cellular Automata", APDCM Workshop, Proc. IEEE IPDPS'04, Santa Fe, New Mexico, 2004
20. Stephen Wolfram "Twenty problems in the theory of CA", Physica Scripta 9, 1985
21. Stephen Wolfram (ed.), "Theory and applications of CA", World Scientific, Singapore, 1986
22. Stephen Wolfram, "Cellular Automata and Complexity (collected papers)", Addison-Wesley, 1994
23. Stephen Wolfram, "A New Kind of Science", Wolfram Media, Inc., 2002