PHYSICAL REVIEW E VOLUME 57, NUMBER 3 MARCH 1998

Computing with cellular automata: Three cases for nonuniformity

Moshe Sipper*
Logic Systems Laboratory, Swiss Federal Institute of Technology, IN-Ecublens, CH-1015 Lausanne, Switzerland

(Received 11 June 1997; revised manuscript received 6 October 1997)

Recently, there has been a resurgence of interest in the use of cellular automata (CA) as computational devices. This paper demonstrates the advantages of nonuniform CAs, in which cellular rules may be heterogeneous, over the classical, uniform model. We address three problems that require global computation: parity, symmetry, and synchronization, showing that: (1) there does not exist a uniform, radius $r=1$ CA that effectively computes a solution, while (2) construction of a nonuniform CA is straightforward. [S1063-651X(98)04503-6]

PACS number(s): 89.80.+h, 02.70.Rw, 07.05.Bx

Cellular automata (CA) are discrete, dynamical systems that perform computations in a distributed fashion on a spatially extended grid. A cellular automaton consists of an array of cells, each of which can be in one of a finite number of possible states, updated synchronously in discrete time steps according to a local, identical interaction rule.

Since their inception CAs have been used as a formal model for studying phenomena of interest in several scientific fields, including physics, biology, and computer science, to mention but a few. In recent years there has been growing interest in the utilization of CAs as actual computing devices. This is largely due to their ''attractive'' properties where computation is concerned, namely, massive parallelism, locality of cellular interactions, and simplicity of basic components (cells). These greatly facilitate the implementation of CAs as electronic hardware [1,2]. CAs also suggest a possible approach to attaining novel computational architectures at the nanometer scale [3].

A major problem with such highly local systems is the difficulty in designing them to exhibit a specific behavior or solve a particular problem. This results from the local dynamics of the system, which renders the design of local interaction rules to perform global computational tasks extremely arduous. It is important to understand what we mean by this latter term ''global computation.'' A local computation involves the calculation of a property that can be expressed in purely local terms, i.e., as a function of the local cellular neighborhood. Such local operators abound, for example, in the domain of low-level image processing, where various filters and noise-reducing operations are defined in terms of the cell's nearest neighbors [4]. By global computation we mean calculation of a nonlocal property, i.e., one that cannot be defined in purely local terms. The difference between global and local computation is not a hard distinction, but a matter of degree. Some properties may be only slightly nonlocal, requiring but a small extension beyond the cell's local neighborhood, while others may be highly global, in the sense that the entire grid must be considered in order to compute a correct output.

The distinction between global and local computation made in the previous paragraph is by no means intended to serve as a formal definition. Rather, it aims at motivating the question of computation in CAs, addressed in this paper. Our intention is to investigate a number of problems that involve global computation, where the CA must compute a given property that is ''distributed'' over the entire grid.

One of the attributes of the CA model is its uniformity, meaning that the cellular rule is identical for all cells. We have recently been investigating a variation of this model, namely, nonuniform CAs, where cellular rules need not be identical [1,5,6]. This research is centered on the application of an evolutionary algorithm, known as cellular programming, to evolve nonuniform CAs to perform computations. Our results to date suggest that this model exhibits interesting behavior in general, offering, in particular, new paths in the pursuit of novel computational architectures. Nonuniform CAs seem to have received very little attention in the literature (one of the few references we were able to trace is [7]), though from a hardware point of view they offer the same aforementioned advantages of uniform CAs. In a recent work, Benjamin and Johnson [3] presented a one-dimensional CA that can perform binary addition. In fact, the CA in question is nonuniform, consisting of two distinct rules, which are supplemented by a third one in order to enable implementation using coupled quantum dots [8].

Increasing the problem-solving capacities of a CA can come about by modifying parameters other than the uniformity of rules, such as the number of states per cell or the cellular radius (or both). However, these usually entail a much higher cost where implementation is concerned since the resulting system exhibits denser connectivity and more complex cells. This is true of nanometer-scale machines as well as more conventional electronic hardware. It can also be observed that a finite number of different rules in a nonuniform CA can be transformed into one single uniform rule with more states — again, this entails a high implementation cost.

The goal of this paper is to demonstrate the advantages of nonuniformity by addressing three problems that require global computation: parity, symmetry, and synchronization. For each of these problems we show that: (1) there does not exist a uniform radius $r=1$ CA that effectively computes a solution, while (2) construction of a nonuniform CA is straightforward. By CA solution we mean one that applies to any grid size N. When considering CAs that perform computa-

*Electronic address: Moshe.Sipper@di.epfl.ch tions it is understood that the input to the computation is encoded as an initial state configuration, and the output is the configuration after a certain number of time steps.

The CAs studied herein are binary state, with radius $r=1$ (i.e., each cell is connected only to two other cells, those to its immediate left and right). A cell's rule table is specified by 8 bits that determine the cell's state at time step $t+1$ as a function of its state along with those of its two neighbors at time step t:

t	111	110	101	100	011	010	001	000
$t+1$	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0

where $r_i \in \{0,1\}$, $i=0,\ldots,7$. In accordance with Wolfram's standard notation [9], a rule is denoted by a decimal value between 0 and 255, equal to $\Sigma_{i=0}^{7} 2^i r_i$.

The parity problem. In this problem the initial (input) configuration must be classified into two classes, according to whether it contains an odd number of ones or not (the output is thus the parity of the input — odd or even). The problem was given a good deal of attention in Minsky and Papert's seminal work in the field of artificial neural networks [10]. The problem's difficulty stems from the fact that the most similar patterns (those which differ by a single bit) require different outputs. It was proven to be a hard problem for a single-layer perceptron [10], and treated several years later in another seminal work, that of Rumelhart, Hinton, and Williams [11], who showed that multilayer perceptrons can be trained to solve it. The parity problem still remains an important benchmark where neural networks are concerned, and our aim below is to treat it within the context of cellular automata.

We first proceed to show that a uniform, binary-state, $r=1$ CA cannot solve this problem. Our demonstration is based upon the following observation: assume that a CA must classify any input configuration into one of n distinct classes, where each configuration belongs to exactly one class. Then, for a configuration belonging to class i, at no point during the CA's time evolution can the grid configuration consist of a pattern belonging to class j, $j \neq i$ $(i,j \in \{1,\ldots,n\})$. This means that upon giving an input configuration belonging to class i, the CA must pass through a state trajectory consisting solely of class-i configurations. The reason is apparent — if this class-preservation rule is violated, i.e., the CA passes through a class-j configuration while treating a class-i pattern $(j \neq i)$, then there exist at least two configurations, belonging to two different classes, which will be classified incorrectly as being of the same class.

Thus, a necessary (though not sufficient) condition that a parity classifier must meet is that it be parity preserving. Below, we "narrow down" the number of possible uniform, $r=1$ CAs that are candidates for solving the parity problem by looking at the constraints on the rule table imposed by the parity-preservation requirement. Null boundary conditions are assumed, i.e., the extreme left and right cells are considered to have a cell with a constant state of zero as their "missing" neighbor. (Note: in what follows the constraints on N may differ between cases, as indicated.)

Constraint 1: $r_0=0$. This results from the fact that given a grid of size N, where N is odd, the configuration 0^N must be mapped to 0^N (mapping it to 1^N would result in a change of class).

Constraint 2: $r_5=0$. Consider the configuration 1^N, where $N \geqslant 3$ is odd. Parity preservation implies (1) $r_7=0 \Leftrightarrow r_3 \neq r_6$, and (2) $r_7=1 \Leftrightarrow r_3=r_6$. Next, consider the configurations $X11111Y$ and $X11011Y$, $X,Y \in \{0,1\}^*$, which obviously belong to two different classes. Thus, at the next time step the CA must maintain an odd (Hamming) distance between the configurations obtained via application of the rule, implying (1) $r_7=0 \Rightarrow r_3 \neq r_6 \Rightarrow r_5=0$, and (2) $r_7=1 \Rightarrow r_3=r_6 \Rightarrow r_5=0$. Thus, we obtain the constraint $r_5=0$.

Constraint 3: $r_2=1$. Consider the configuration $\{10\}^M 1$, of size $N=2M+1$, where M is even. Since we have already established that $r_5=0$, parity preservation requires that $r_2=1$.

Constraint 4: $r_1=r_4=0$. Consider the configuration $0^{N-1}1$, $N \geqslant 2$. Since $r_0=0$ and $r_2=1$ this requires that $r_1=0$. Similarly, considering the configuration 10^{N-1} results in $r_4=0$.

Constraint 5: $r_3=r_6$. Consider the configuration $01^{N-2}0$, where $N \geqslant 4$ is even. Since $r_1=r_4=0$, this implies that $r_3=r_6$.

Constraint 6: $r_7=1$. This follows from constraint 5 $(r_3=r_6)$ and the argument given for constraint 2 $(r_7=1 \Leftrightarrow r_3=r_6)$.

The above constraints narrow the possible parity classifiers to two rules: 132 and 204. This seems to be the best one can do, since the values of r_3 and r_6 cannot be further delimited — if either neighborhood occurs somewhere along the grid, the other must occur as well (due to the null boundary conditions). Their equality implies that parity is preserved, whatever the actual value.

Rule 204 is the identity rule, i.e., one that transforms the state of a cell to itself, and can therefore obviously not be considered as an effective parity classifier. Rule 132 seems to offer only a slight improvement over rule 204, settling after a few time steps to one of a large number of fixed points (this is demonstrated exhaustively for small-size grids in Wolfram's book [9], where several tables in the end provide a comprehensive "guide" to uniform, $r=1$ CAs). The only apparent difference from rule 204 is that it slightly reduces the number of 1's in the initial configuration. However, there seems to be no succinct output specification that would indicate the input pattern's class, and the CA cannot therefore effectively compute a solution. An interesting example of a uniform CA solving a hard problem, known as density classification, was recently demonstrated by Capcarrere, Sipper, and Tomassini [12]. This CA (as well as the nonuniform ones discussed herein) performs effective computation in the sense that the complexity of the output is reduced with respect to the input, i.e., "reading" the output is in some sense easier (e.g., with respect to Kolmogorov or computational complexity — see [12]).

If one considers nonuniform CAs parity classification becomes straightforward. Two possibilities would be (1) a grid consisting of rule 240 in all cells, except for the rightmost cell with rule 60. Essentially, rule 240 implements a right shift, while rule 60 performs the XOR operation between its central and left bits. Initializing the rightmost cell to zero, and the rest of the grid with the input configuration, then after $N-1$ time steps the rightmost cell will contain the out-

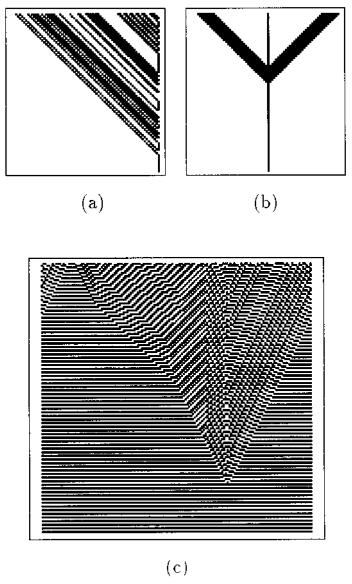
FIG. 1. Demonstration of nonuniform, $r=1$ CAs discussed in text. White squares represent cells in state 0, black squares represent cells in state 1. The pattern of configurations is shown through time, which increases down the page. Null boundary conditions are used for (a) and (b), and periodic boundary conditions for (c). Grid sizes are $N=75$ for (a), (b), and $N=149$ for (c). (a) Parity classifier consisting of rule 240 in all cells, except for the rightmost cell with rule 60. Initial configuration has odd parity. (b) Symmetry classifier consisting of a grid with rule 132 (central cell), rule 240 (all cells to the left), and rule 170 (all cells to the right). Initial configuration is symmetric. (c) A nonuniform CA that solves the synchronization problem, evolved via the cellular programming evolutionary algorithm.

put: a state of 1 for odd parity and 0 otherwise [Fig. 1(a)]. (2) A grid of size $2M+1$, consisting of a central cell with rule 150 (which performs a sum modulo 2 of its three neighboring states), M left cells with rule 240 (right shift), and M right cells with rule 170 (left shift). The central cell, initialized to zero, will contain the output after M time steps.

The symmetry problem. In this problem the even-size input configuration must be classified into two classes, according to whether it is symmetric about its center or not. The problem was discussed by Minsky and Papert [10] within a neural network context, with backpropagation being applied to its solution several years later [11].

Below, we narrow down the number of possible uniform, $r=1$ CAs that are candidates for solving the symmetry problem by looking at the constraints on the rule table imposed by the symmetry-preservation requirement (by this we mean the preservation of a pattern's symmetry or nonsymmetry characteristic, as may be the case). Null boundary conditions are assumed. (Symmetry considerations in CA design were also discussed in [13].)

Constraint 1: $r_1=r_4$, $r_3=r_6$. This results from the fact that a symmetric pattern at time step t must remain so at time step $t+1$, entailing the above symmetries in the rule table.

Constraint 2: $r_1\neq r_2$. Consider the $N=2$ grid configuration 01, transformed in the next time step to r_1r_2. Preservation of its nonsymmetry implies the above constraint.

Constraint 3: $r_3\neq r_4$ *or* $r_7\neq r_6$, where *or* is the logical OR function. Consider the $N=4$ configuration 1110, transformed in the next time step to $r_3r_7r_6r_4$. Preservation of its nonsymmetry implies the above constraint.

Constraint 4: $r_0\neq r_6$ *or* $r_1\neq r_3$. Consider the $N=4$ configuration 0011, transformed in the next time step to $r_0r_1r_3r_6$. Preservation of its nonsymmetry implies the above constraint.

Constraint 5: $r_2\neq r_6$ *or* $r_5\neq r_3$. Consider the $N=4$ configuration 1011, transformed in the next time step to $r_2r_5r_3r_6$. Preservation of its nonsymmetry implies the above constraint.

Constraint 6: $r_0\neq r_2$ *or* $r_1\neq r_5$ *or* $r_3\neq r_6$. Consider the $N=6$ configuration 001101, transformed in the next time step to $r_0r_1r_3r_6r_5r_2$. Preservation of its nonsymmetry implies the above constraint.

Constraint 7: $r_2\neq r_6$ *or* $r_5\neq r_7$ *or* $r_2\neq r_3$ *or* $r_4\neq r_1$. Consider the $N=8$ configuration 10100111, transformed in the next time step to $r_2r_5r_2r_4r_1r_3r_7r_6$. Preservation of its nonsymmetry implies the above constraint.

The above constraints narrow the possible symmetry classifiers to four rules: 51, 90, 165, and 204. Of these rules 204 (identity) and 51 (inverse identity) can immediately be ruled out as possible effective symmetry classifiers. Rules 90 and 165 belong to Wolfram's class 3, consisting of rules that exhibit chaotic, aperiodic behavior (note that these two rules are equivalent under the conjugation transformation, i.e., interchange of the roles of 0 and 1) [9]. We maintain that these cannot perform effective classification, as there seems to be no way to define a succinct output specification that would indicate the input pattern's class.

Constructing a nonuniform CA symmetry classifier is straightforward. A grid of size $2M+1$, consisting of a central cell with rule 132, M left cells with rule 240 (right shift), and M right cells with rule 170 (left shift) can solve the problem. The central cell, initialized to 1, will remain in this state as long as its left and right neighbors match. Upon the arrival of a nonmatching pair, the central cell changes its state to zero, and remains so, signifying a nonsymmetric pattern. If no such pair arrives during M time steps the central-cell state remains 1, signifying a symmetric pattern [Fig. 1(b)].

The synchronization problem. In this problem the CA, given any initial configuration, must reach a final configuration, within a fixed number of time steps, that oscillates between all 0's and all 1's on successive time steps. Spatially periodic boundary conditions are used, resulting in a circular grid. The problem was introduced by Das *et al.* [14] who applied a genetic algorithm to evolve uniform CAs to solve it. They noted that the task is nontrivial since synchronous oscillation is a global property of a configuration, whereas a small-radius CA employs only local interactions. Thus, while local regions of synchrony can be directly attained, it is more difficult to design CAs in which spatially distant regions are in phase. Since out-of-phase regions can be distributed throughout the grid, transfer of information must occur over large distances [i.e., $O(N)$] to remove these phase defects and produce a globally synchronous configuration.

The synchronization problem is different from the previous two in that the CA's ultimate behavior is specified (a period-2 cycle) rather than a classification criterion on the input patterns. Thus, one can easily show by empirical means that no uniform $r=1$ CA exists that can solve this problem.

Wolfram [9] provides a complete listing of all cycles for CAs of sizes $N=9,10,11$ (these are small enough to yield to exhaustive testing of all possible input configurations) — examination of these tables reveals that no $r=1$ CA exhibits precisely one or two cycles of period two. We have performed tests on larger grids (statistical rather than exhaustive), confirming this result [1]. Clearly, the synchronization problem cannot be solved by a uniform $r=1$ CA (at least not for all grid sizes N).

Constructing a nonuniform CA to solve the synchronization problem is again straightforward. A grid consisting of a left cell with rule 51, with all other cells containing rule 15 can solve the problem. Essentially, rule 51 simply flips its state at each time step, with rule 15 entering the opposite state of its left neighbor. Synchronization will be attained after N time steps. Another solution, exhibiting more ''intricate'' dynamical behavior, was evolved by the cellular programming evolutionary algorithm [1], and is demonstrated in Fig. 1(c) (a formal framework addressing the meaning of ''intricateness'' of a CA's dynamical behavior is outlined in [15]).

An interesting observation concerning the nonuniform systems presented herein is their high degree of ''regularity'' — while a nonuniform CA can possibly contain a plethora of different rules we have found that only a small number may be needed in practice. Such CAs, dubbed quasiuniform, were also observed to emerge via cellular programming [1].

Continued increase in computing power will someday (perhaps soon) require novel computational architectures. One emerging approach seems to be that of CA-like systems. We have endeavored to demonstrate above that departing from the classical model, changing, as it were, the rules of the game, offers promising future prospects.

The author is grateful to Marco Tomassini for his helpful comments.

[1] M. Sipper, *Evolution of Parallel Cellular Machines: The Cellular Programming Approach* (Springer-Verlag, Heidelberg, 1997).

[2] M. Sipper, E. Sanchez, D. Mange, M. Tomassini, A. Pérez-Uribe, and A. Stauffer, IEEE Trans. Evolutionary Comput. **1**, 83 (1997).

[3] S. C. Benjamin and N. F. Johnson, Appl. Phys. Lett. **70**, 2321 (1997).

[4] A. Broggi, V. D'Andrea, and G. Destri, Int. J. Mod. Phys. C **4**, 5 (1993).

[5] M. Sipper, Physica D **92**, 193 (1996).

[6] M. Sipper, in *Annual Reviews of Computational Physics*, edited by D. Stauffer (World Scientific, Singapore, 1997), Vol. V, pp. 243–285.

[7] G. Y. Vichniac, P. Tamayo, and H. Hartman, J. Stat. Phys. **45**, 875 (1986).

[8] A. O. Orlov, I. Amlani, G. H. Bernstein, C. S. Lent, and G. L. Snider, Science **277**, 928 (1997).

[9] S. Wolfram, *Cellular Automata and Complexity* (Addison-Wesley, Reading, MA, 1994).

[10] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry* (MIT Press, Cambridge, MA, 1969).

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, in *Parallel Distributed Processing, Volume 1: Foundations*, edited by D. E. Rumelhart, J. L. McClelland, and the PDP Research Group (The MIT Press, Cambridge, MA., 1986), pp. 318–362.

[12] M. S. Capcarrere, M. Sipper, and M. Tomassini, Phys. Rev. Lett. **77**, 4969 (1996).

[13] U. Frisch, B. Hasslacher, and Y. Pomeau, Phys. Rev. Lett. **56**, 1505 (1986).

[14] R. Das, J. P. Crutchfield, M. Mitchell, and J. E. Hanson, in *Proceedings of the Sixth International Conference on Genetic Algorithms*, edited by L. J. Eshelman (Morgan Kaufmann, San Francisco, CA, 1995), pp. 336–343.

[15] J. P. Crutchfield and M. Mitchell, Proc. Natl. Acad. Sci. USA **92**, 10 742 (1995).