Bull. Nov. Comp. Center, Comp. Science, 33 (2012), 1–9
© 2012 NCC Publisher

On the reduction of computational complexity of cellular automata

Mikhail Abramskiy

Abstract. The computational complexity of cellular automata (CA) is investigated. Using unified approach to the CA behavior, we define the notion of CA convergence and propose the measures of the time and space complexity. The approach to the complexity reduction for some classes of synchronous CA is discussed. Then we consider the counterexample that shows that our reduction methods are not, in general, applicable to asynchronous models of CA.

1. Introduction

Cellular automata (CA) were invented in the 1950s by Von Neumann, and nowadays they remain a very popular subject of research. The reasons of such an interest lay in intrinisic properties of CA such as fine-grained parallelism, time and space homogeneity, a simple way of definition and large computational power. The CA are suitable to many research fields. CA models were applied to several simulation problems such as diffusion, self-organization, wave propagation, etc. [1]. They are also considered as a discrete alternative for partial differential equations because of the possibility of simulating non-linear processes [3]. Any CA simulation requires two types of computational resources: time and space. Despite the growth of productivity of computers and wide opportunities for parallel implementation of the CA, the question of reducing the computational costs is still actual. Another problem is to find suitable complexity measures that would fully represent intrinsic properties of the CA. The complexity of CA was studied by Wolfram in the 1980s. He proposed the classification of the CA behavior complexity in [2]. An approach to connect the behavior complexity and the computational complexity of CA was presented in [4]. This paper is an attempt to give a unified approach to the definition of space and time complexity of CA and classifying its behavior according to the computational complexity. In Section 2, we introduce necessary definitions of a CA model. In Section 3, we introduce a unified approach to the CA behavior and define the notion of CA complexity and convergence. In Section 4, we show how the time and space complexity of the CA for some classes of synchronous CA can be reduced. In Section 5, we discuss the applicability of our complexity reduction approach to the asynchronous CA model.

2. Cellular automata model

Let $\mathbb{Z}$ be a set of integer numbers. Consider the discrete n-dimensional space with integer vectors:

$$\mathbb{Z}^n = \{(x_1, x_2, \dots, x_n) \mid x_i \in \mathbb{Z}\}.$$

The *naming set*

$$M = \big\{(x_1, x_2, \dots, x_n) \mid x_i \in \{0, 1, \dots, L_i - 1\}\big\}$$

is a finite subset of $\mathbb{Z}^n$. We will use m for its elements instead of writing $(x_1, x_2, \dots, x_n)$. Note that $|M| = L_1 L_2 \cdots L_n$.

Let $A = \{a_1, a_2, \dots, a_k\}$ be a finite state alphabet. A pair $\langle a, m\rangle$, where $a \in A$, $m \in M$, is called a *cell.* In general, the sets A and M form a *cellular array* $\Omega(A, M) = \{\langle a, m\rangle \mid a \in A,\ m \in M\}$ — a set of cells, that is also called a *global configuration.*

The next step is to define a *neighborhood. Naming functions* $\phi : M \to M$ are defined on M. The most widespread template for ϕ is

$$(x_1 + b_1, x_2 + b_2, \dots, x_n + b_n), \quad b_i \in \{-r, \dots, 0, \dots r\}.$$

If $m' = \phi(m)$, $m' \in M$, then m' is a *neighbor* for m. A set

$$T(m) = \{\phi_0(m), \phi_1(m), \dots, \phi_d(m)\}$$

is called a *neighborhood template* or just a *neighborhood* for m, and the positive number r is the *radius* of the neighborhood. Usually $\phi_0(m) = m$. Sometimes we present a neighborhood template by the set of shifts

$$\{(0, 0, \dots, 0), (b_{11}, b_{12}, \dots, b_{1n}), \dots, (b_{d1}, b_{d2}, \dots, d_{dn})\}.$$

One of the most widespread neighborhood templates is the Moore neighborhood $\{(-1,-1),\ (-1,0),\ (-1,1),\ (0,-1),\ (0,0),\ (0,1),\ (1,-1),\ (1,0),\ (1,1)\}$ (a square with the side equal to 3). Further we will use it in examples.

A set $S(m) = \{(a_{i_0}, m), (a_{i_1}, \phi_1(m)), \dots, (a_{i_d}, \phi_d(m))\}$ is called a *local configuration* and the corresponding vector $(a_{i_0}, a_{i_1}, \dots, a_{i_d})$ is a *state vector.* The *local transition function* $\theta : A^{d+1} \to A$ is defined on the state vectors and its value a' is the element from A to be placed in the cell (a, m) instead of a.

Finally, we can give the definition of a CA model. The *cellular automaton* is a tuple $\aleph = \langle A, M, T, \theta\rangle$, where A is an alphabet, M is a naming set, T is a neighborhood template, and θ is a local transition function.

We will call a *step* a single application of θ to one cell. Applying θ to all the cells from the cellular array Ω, we perform the *iteration* that can be described by the *global transition* $\Theta : \Omega(t) \to \Omega(t+1)$. A sequence Ω, $\Omega(1)$, $\Omega(2)$, $\ldots$, $\Omega(t)$, $\ldots$ is called the *evolution* of CA. Here t is the iteration number and $\Omega(t)$ stands for the global configuration on the iteration t.

Let $\langle a, m \rangle$ be a cell. We use the notion $a = \aleph(m,t)$ as the state of the cell of the automaton $\aleph$ with the name m on the iteration t. We will use $\aleph(m)$ for $\aleph(m,0)$.

A necessary thing to discuss is the mode of CA evolution. Synchronous and asynchronous are the most common modes. In the *synchronous mode* cells update their states only after all their subsequent states are computed. On a sequential computer, this mode can be simulated by storing the new states obtained in memory and then updating all of them after $|M|$ computations of the new cell states are completed. The *asynchronous mode* suggests an immediate update of a cell after obtaining its new state. So, the global transition (iteration) of asynchronous CA consists of $|M|$ sequential acts of cell updating. The order of these acts is, generally, random, but there exists an ordered asynchronous CA, that allows updating only in a fixed order.

Finally, we have to fix boundary conditions of CA. Consider a 1D CA with $M = \{0, 1, 2\}$ and the neighborhood template $(-1, 0, 1)$. This means that arguments of the local transition function θ are the states of a cell and its left and right neighbors. But what will happen if we apply θ to the cell 0 that actually doesn't have the left neighbor or to the cell 2 that doesn't have the right neighbor? And, in general, if $T(m) = \{\phi_0(m), \phi_1(m), \ldots, \phi_d(m)\}$ is a neighborhood template with a radius r, what happens if $\phi_i(m) \notin M$?

We deal with this problem using the *periodic boundary conditions*. As was mentioned before, $\phi_i(m) = (x_1 + b_{i1}, x_2 + b_{i2}, \ldots, x_n + b_{in})$, $b_{ij} \in \{-r, \ldots, 0, \ldots r\}$. Thus, the described problem arises when for some j we have $x_j + b_{ij} < 0$ or $x_j + b_{ij} \geq u_j$. We use the modulo operation to return values to the required bounds:

$$0 \leq (x_j + b_{ij}) \bmod u_j \leq u_j - 1, \quad \forall j = 0, 1, \ldots, d$$

Such an operation is correct since $x_j + b_{ij} = (x_j + b_{ij}) \bmod u_j$ for $0 \leq x_j + b_{ij} \leq u_j - 1$.

In the example described we have $(0-1) \bmod 3 = 2$ for the letmost cell. This means that we assume the rightmost cell to be the left neighbor of the leftmost cell. Respectively, the leftmost cell is assumed to be the right neighbor of the rightmost cell, i.e., $(2+1) \bmod 3 = 0$.

3. The wolfram behavior and computational complexity

Before consideration of the complexity characteristics of the CA, we should find a general approach to describe the CA behavior. The global behavior characterization of CA was described by Stephen Wolfram. In [2], he proposed four classes of the CA behavior based on the stable states of the CA evolution:

Class 1: homogeneous global state,

Class 2: periodic behavior,

Class 3: complex structures, and

Class 4: chaotic behavior.

This classification describes the behavior of a general class of CA with a potentially infinite naming set. But in our paper, we deal only with CA with a finite naming set (finite lattice). Hence, we have the number of different CA global configuration to be finite and equal to $|M|^{|A|}$. As soon as our local transition function θ is deterministic, the global transition Θ is deterministic. Therefore we will not have any chaotic or complex behavior (Classes 3 and 4) in the CA model described.

The periodic behavior of CA (Class 2) can be described as follows: on the iteration t_0, we have a configuration that appeared T iterations before, i.e., $\Omega(t_0) = \Omega(t_0 - T)$. According to deterministic properties of the global transition we have $\Omega(t_0+k) = \Omega(t_0-T+k)$, $k = 1, 2, \ldots$ Thus, we can state that a CA attains a periodic behavior on the iteration t_0 with the period T.

The CA of Class 1 (homogeneous global state) can be described in the same terms having $T = 1$. In other words, the homogeneous global state means that for iterations $t \geq t_0$, we have $\Omega(t + 1) = \Omega(t)$. Such a behavior can be assumed to be periodic from the iteration t_0 with the period $T = 1$.

So, any CA with a finite set reaches the periodic behavior. The CA *converges* if its period T is equal to 1.

Using this unified description of the CA behavior, we can now speak about the computational complexity of the CA model. It is not difficult to see that there are two general complexity measures that can be applicable to the CA — space and time.

Let $\aleph = \langle A, M, T, \theta \rangle$ be a CA. Speaking about the space complexity of $\aleph$, we will refer to the number of cells, i.e., to the power of the naming set M. So, the space complexity is $S(\aleph) = |M| = u_1 u_2 \cdots u_n$.

The naive way to define the time complexity of a CA is to describe it by the number of iterations resulting in the periodic behavior. However, the iteration itself consists of $|M|$ steps and at each step, the computation of the value of the local transition function θ is performed.

So, we actually have two measures of the time complexity. We will apply $T_i(\aleph)$ considering the number of iterations and $T_s(\aleph)$ — in the second case.

Let $\tau(\theta)$ be the time complexity of θ described in some common way, for example, Turing Machine steps or Boolean circuit size. Then we have

$$T_s(\aleph) = T_i(\aleph)\, S(\aleph)\, \tau(\theta). \qquad (1)$$

4. Complexity reduction for CA

Let us consider two one-dimensional CA with Boolean alphabet, the same neighborhood template T and the local transition function $\theta(x, y, z) = x \wedge \bar{y} \wedge z$, but with different naming sets M_1 and M_2 with $|M_1| = 9$ and $|M_2| = 3$. Both CAs are functioning in the synchronous mode with the periodic boundary conditions. Automata start their operation from global configurations, respectively, 101101101 and 101. The evolution is shown in Table 1.

Table 1

t	$\Omega_1(t)$	$\Omega_2(t)$
0	101101101	101
1	010010010	010
2	000000000	000
3	000000000	000

We can see that on every iteration t, the global configuration $\Omega_1(t)$ of the first CA can be described as the concatenation $\Omega_2(t)\Omega_2(t)\Omega_2(t)$. It is not difficult to check that this fact takes place every time when the initial global configuration of the first CA Ω_1 is presented as concatenation $\Omega_2\Omega_2\Omega_2$.

So, we can see that if the global configuration of CA can be partitioned into identical parts, we can use instead another CA with reduced space complexity that processes on one partition, shows the same convergence behavior and gives on every iteration the global configuration that can be translated into the global configuration of the original CA by repeating it the necessary number of times in several dimensions.

We are going to prove this fact in the general case for every n-dimensional synchronious CA with a finite alphabet with periodic boundary conditions.

Consider two CAs

$$\aleph_1 = \langle A, M_1, T, \theta \rangle, \qquad |M_1| = L_1 L_2 \cdots L_n,$$
$$\aleph_2 = \langle A, M_2, T, \theta \rangle, \qquad |M_2| = K_1 K_2 \cdots K_n.$$

Numbers $\{L_i\}$ and $\{K_i\}$ are related as follows:

$$L_i = K_i w_i, \quad w_i \geq 1, \quad i = 1, 2, \ldots, n, \qquad \prod_{i=1}^{n} w_i > 1.$$

We initialize a cellular array of $\aleph_2$ randomly with elements from A. And then we construct the initial global configuration for $\aleph_1$ by the rule

$$\aleph_1(x_1 + c_1 K_1, \ldots, x_n + c_n K_n) = \aleph_2(x_1, \ldots, x_n), \tag{2}$$
$$0 \leq x_i \leq K_i - 1, \quad 0 \leq c_i \leq w_i - 1.$$

Further we will use m instead of $(x_1, \ldots, x_n)$ and $m + C$ instead of $(x_1 + c_1 K_1, \ldots, x_n + c_n K_n)$.

Theorem. *The configuration of $\aleph_1$ can be constructed from $\aleph_2$ on every iteration t using (2), i.e., $\aleph_1(m + C, t) = \aleph_2(m, t)$ for any iteration t and $0 \leq x_i \leq K_i - 1$, $0 \leq c_i \leq w_i - 1$.*

Proof. Let us prove the theorem by induction on the number of iterations of $\aleph_1$.

Basis: $t = 0$. Follows directly from (2).

Induction step: We assume that $\aleph_1(m + C, t_0) = \aleph_2(m, t_0)$ for $t = t_0$. Note that the neighborhood template T of $\aleph_1$ is $\{m, \phi_1(m), \dots, \phi_d(m)\}$. For iteration $t = t_0 + 1$ we have:

$$\begin{aligned}\aleph_1(m + C, t_0 + 1) &= \theta(\aleph_1(m + C, t_0), \dots, \aleph_1(\phi_d(m + C), t_0))\\ &= \theta(\aleph_2(m, t_0), \dots, \aleph_2(\phi_d(m), t_0))\\ &= \aleph_2(m, t_0 + 1).\end{aligned}$$ □

Corollary. $T_i(\aleph_1) = T_i(\aleph_2)$.

Proof. In Section 3, we have shown that every CA model converges to a periodic behavior. Let $\aleph_2$ have reached the periodic behavior on the iteration t_0 with the period T. In other words, $\aleph_2(m, t + T) = \aleph_2(m, t)$.

At the same time, from the theorem we have

$$\aleph_1(m + C, t + T) = \aleph_2(m, t + T) = \aleph_2(m, t) = \aleph_1(m + C, t).$$

So, $\aleph_2$ also reaches the periodic behavior on t_0 with the period T. Thus, automata have an equal time complexity in terms of iterations. □

It is easy to see from the construction of $\aleph_1$ and $\aleph_2$ that

$$S(\aleph_1) = S(\aleph_2) \cdot w_1 w_2 \cdots w_n,$$

and, therefore, from the corollary and (1) we have

$$T_s(\aleph_1) = T_s(\aleph_2) \cdot w_1 w_2 \cdots w_n.$$

Therefore, having the time complexity in terms of iterations equal in both cases, we have essentially reduced the space complexity and the time complexity in terms of steps.

Let us see an example how this theorem works. Consider two-dimensional CAs $\aleph_1$ and $\aleph_2$ with Boolean alphabet, square lattices of site, respectively, 100×100 and 20×20, the moore neighborhood and the transition function

$$\theta(m) = \begin{cases} 1 & \text{if} \quad \sum\limits_{T_M} a_i > 5 \quad \text{or} \quad \sum\limits_{T_M} a_i = 4,\\ 0 & \text{otherwise,}\end{cases}$$

where $\sum_{T_M} a_i$ is the sum of states in the Moore neighborhood of a cell m.

This function has been taken from the phase separation CA. Both $\aleph_1$ and $\aleph_2$ work with periodic boundary conditions. We randomly generate the initial global configuration for $\aleph_2$ and then spread it to $\aleph_1$ using the process described above (Figure 1).

When we run a CA we can see that the configurations of $\aleph_1$ and $\aleph_2$ do not contradict with the theorem.

The configuration on iterations 20, 40, 60 and 80 is shown in Figure 2.

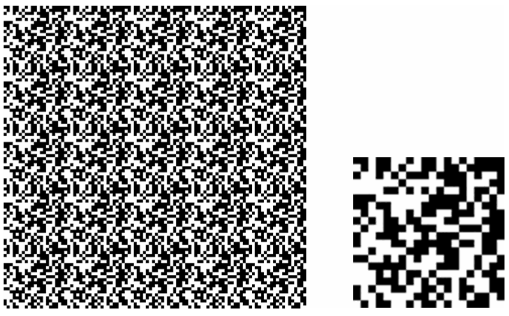

Figure 1. Initial configurations of $\aleph_1$ (on the left) and $\aleph_2$ (on the right)

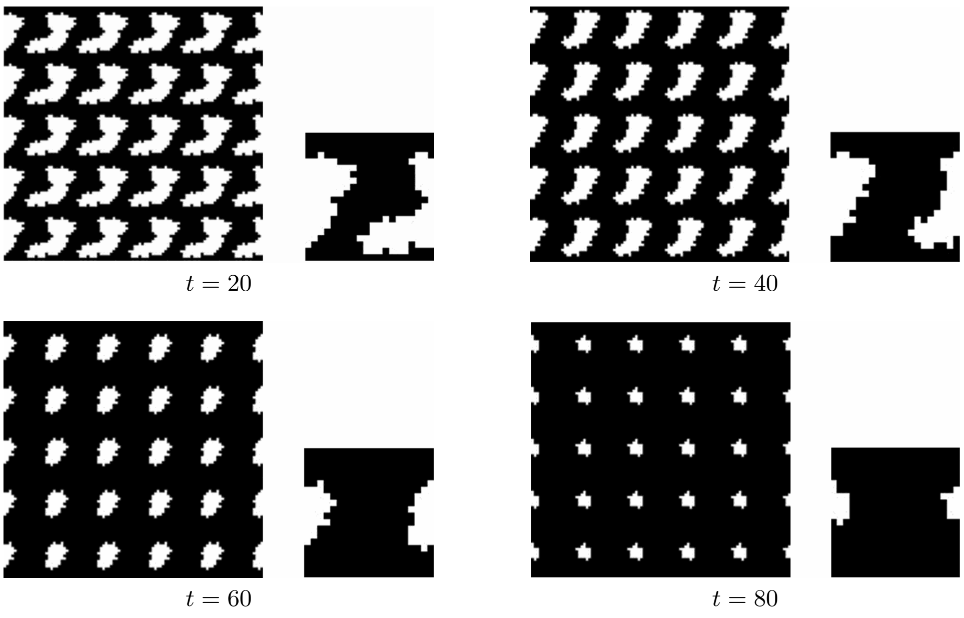

Figure 2. Snapshots of $\aleph_1$ and $\aleph_2$ evolutions at iterations $t = 20$, 40, 60, and 80

5. Asynchronous CA

Consider two CAs

$$\daleth_1 = \langle (0,1), \{0,1\}, \{0,1\}, \theta \rangle, \qquad \daleth_2 = \langle (0,1), \{0,1,2,3\}, \{0,1\}, \theta \rangle$$

with initial configurations 01 and 0101, respectively, that work in asynchronous mode. We will use the logical implication $\to$ as the transition function θ.

As we can see, all necessary conditions for the theorem are satisfied. We will now show that the behavior of automata will be in contradiction with the theorem after the first iteration.

The first iteration of $\beth_1$ consists of the two steps:

- applying a local transition function to the cell 0 and immediate updating of the state of the cell 0,
- the same actions with the cell 1 using the new state of the cell 0.

The evolution is described in Table 2.

Now we will do the first iteration of $\beth_2$ that consists of four steps of applying a local transition function to the cells 0, 1, 2, 3 in random order. Let a random order on the first iteration use the cells 3, 1, 0, 1 (we cannot ensure that we will use every cell on every iteration). The result is described in Table 3.

Table 2. Evolution of $\beth_1$

Step	Iteration	Configuration	Comment
0	0	01	
1	1	11	$0 \to 1 = 1$
2	1	11	$1 \to 1 = 1$

Table 3. Evolution of $\beth_2$

Step	Iteration	Configuration	Comment
0	0	0101	
1	1	0100	$1 \to 0 = 0$
2	1	0000	$1 \to 0 = 0$
3	1	1000	$0 \to 0 = 1$
4	1	1100	$0 \to 0 = 1$

Thus, we found that the theorem does not work for $\beth_2$, so we cannot carry out the described complexity reduction in the general case for asynchronous CA.

6. Conclusion

The results obtained show that we can essentially reduce the computational complexity of synchronous CA using them for simulating or computing problems, that may have initial configurations that can be partitioned in the several parts. These can be the self-organization and self-replication problems, texture recognition problems, etc.

Further research into this subject will be devoted to the model of block-synchronous CA that are commonly used in simulation problems and considered as a trade-off between the virtual parallelism of synchronous CA and the sequential behavior of asynchronous CA.

References

[1] Bandman O. Cellular Automata models of spatial dynamics // System Informatics / A.G. Marchuk, ed.—Inst. Inform. Systems SB RAS, 2006.—Issue 10.—P. 59–114.

[2] Wolfram S. Universality and complexity in cellular automata // Physica D.—1984.—Vol. 10.—P. 1–35.

[3] Toffolli T., Margolus N. Cellular Automata Machines.—MIT Press, USA, 1987.

[4] Di Lena P., Margara L. Computational complexity of dynamic systems: the case of cellular automata // Information and Computation.—2008.—Vol. 206, Issue 9–10.—P. 1104-1116.