Bull. Nov. Comp. Center, Comp. Science, 19 (2003), 1–13

Spatial functions approximation by Boolean arrays

O. Bandman

Since Cellular Automata attract a growing interest as a model to be used in simulating spatial dynamics, the problem arises of Boolean data compatibility with continuous spatial functions widespread in physics. To solve this problem, a method to approximate real functions in discrete space by Boolean arrays and vice versa is proposed. The approximation errors are assessed, and techniques of decreasing them are considered. Arithmetic operations in the domain of Boolean and real arrays are considered.

1. Introduction

Fast development of supercomputers, clusters and special purpose cellular processors brought about the activization of searching for appropriate mathematical means for spatial dynamics simulation, most interest being attracted to discrete models encompassed with the concept of fine-grained parallelism [1, 2]. All of them have their origin in classical Cellular Automaton (CA) and are either its modifications or a certain kind of its extension. The motivation for investigating CA capabilities in spatial dynamics modeling is based on their "complex systems" properties [3]. The term implies a class of systems composed of many very simple identical processors which act in cooperation and exhibit a very complex overall behavior, such that cannot be described by any other mathematical model.

Although CAs have been announced as an alternative to the conventionally used partial differential equations (PDEs) [4], the latter have a fundamental theoretical and methodological basis, and are nowadays the most often used for the spatial dynamics studies. Undoubtedly, such a situation will remain for a long time. Meanwhile, the scope of CA-modeling is growing fast. There are now many physical, chemical, biological and other phenomena, whose CA-models are known and well studied. For some of them it is proved that they have certain advantages as compared to PDEs, being computationally stable and accurate, easy to program, and admitting any kind of parallel realization [3]. Naturally, the problem arises of making the CA approach compatible with mathematical physics. First, an interface is needed between the usual continuous spatial functions and their representation in the form of Boolean arrays used in CA. Moreover, such an interface

is urgently required when the process under simulation is to be expressed as a combination of two subprocesses, one of them being represented as CA evolution and another one being given in the form of a real function [5].

The transfer from a continuous spatial function to a Boolean array has two stages: 1) discretization of space, which results in a *real cellular array* – a set of cells with real entries (in numerical methods one deals with a dual of a cellular array, referred to as a "grid"), 2) approximation of a real cellular array by a *Boolean cellular array*, referred to as a *Boolean discretization*. Since the first stage is a well-known procedure used in all numerical methods, it is of no interest here. So, the approximation of a real cellular array by a Boolean one is the subject of the paper, the transform being extended to perform the main arithmetic procedures over the arguments of both data types. Clearly, since the transformation of a Boolean array to a discrete spatial function, as well as the inverse one is approximate, the approximation accuracy is also considered.

Apart from Introduction and Conclusion, the paper has four sections. The second section contains formal definitions and relations between a cellular array with real states and its Boolean discretization. The procedure of Boolean discretization is presented in the third section. In the fourth section a few techniques for accuracy improvement are presented. The fifth section concerns arithmetic procedures on the arrays.

2. Main concepts and definitions

To simulate of natural phenomena in space and time is to compute a function $u(\boldsymbol{x}, t)$, where u is a scalar, representing a certain physical value, which may be pressure, density, velocity, concentration, temperature, etc., t stands for time. A vector $\boldsymbol{x}$ represents a point in a continuous space. In the case of D-dimensional Cartesian space the vector components are its coordinates. For example, with $D = 2$, $\boldsymbol{x} = (x_1, x_2)$. Since, in the sequel, spatial functions are of interest, the function $u(\boldsymbol{x})$ with a fixed t is under consideration.

When numerical methods of PDE solution are used for simulation of spatial dynamics, the space is converted into a discrete grid, which is further referred to as a *cellular space* according to the cellular automata terminology. For the same reason the function $u(\boldsymbol{x})$ is represented in the form of a *cellular array*

$$U(\boldsymbol{R}, M) = \{(u, m) : \ u \in \boldsymbol{R}, \ m \in M\}, \tag{1}$$

which is a set of cells, each being a pair (u, m), where u is a state variable with the domain in a set of real numbers $\boldsymbol{R}$, $m \in M$ is the name of a cell. To indicate the state value of the cell named m the notation $u(m)$ is also used. A set of cell names is called a *naming set*. In practice, the names are given by the cells coordinates in a cellular space. For example, in the case

of the cellular space represented by a 2D Cartesian lattice, the names are $M = \{(i,j):\ i,j = 0,1,2\ldots\}$, where $i = x_1/h_1$, $j = x_2/h_2$, h_1 and h_2 being space discretization steps. For simplicity we take $h_1 = h_2 = h$. In theory, it is more convenient to deal with a generalized notion of the naming set M, considering $m \in M$ as a discrete spatial variable.

A cell named m is called *empty* if its state is zero. A cellular array with all the cells being empty is called an *empty array*. It is further denoted as $\Omega = \{(0,m):\ \forall m \in M\}$.

The most profound discretization of a spatial function is its approximation by a Boolean array

$$V(B,M) = \{(v,m):\ v \in B,\ m \in M\}, \quad B = \{0,1\}. \tag{2}$$

In order that this type of approximation be defined a concept of *averaging* should be introduced. It is based on the following definitions. A set of cells

$$Av(m) = \{(v, \phi_k(m)),\ v \in B,\ k = 0,1,\ldots,q\} \tag{3}$$

is called the *averaging area* of a cell named m with $q = |Av(m)|$ being its cardinality, $\phi_k : M \to M$ is a "naming function", indicating to the name of a cell in the averaging area. In the naming set $M = \{(i,j)\}$, the naming functions are usually in the form of shifts, $\phi_k(i,j) = (i+a, j+b)$, a, b being integers not exceeding a fixed r, called a *radius* of averaging.

The *averaged state* of a cell named m is

$$w(m) = \sum_{k=0}^{q} v(\phi_k(m)). \tag{4}$$

The averaging procedure of a Boolean array $V(B,M)$ is to obtain the cellular array $Av(V) = W(A_w, M)$, where $w(m)$ for each $m \in M$ is computed according to (4), $W(A_w, M)$ being referred to as the averaged form of $V(B,M)$. According to (4), the domain of the state variable $w(m)$ is a finite set of numbers forming a discrete alphabet $A_w = \{0,1,2,\ldots,q\}$.

The relation between the real states in the cells $(u,m) \in U(\boldsymbol{R}, M)$ and the averaged values in the cells $(w,m) \in W(A_w, M)$ depends on the scaling parameters, which are $q = |Av(m)|$ and $u_{\max}$. The latter value comes from the assumption (which is usual in mathematical physics) that the function $u(\boldsymbol{x}, t)$ under simulation is bounded in the simulation domain. This means that at any t and $\boldsymbol{x}$ its value does not exceed a certain value $u_{\max}$. So, the normalized real values $y = u/u_{\max}$ may be used, the *normalized cellular array* being $\mathrm{Norm}(U) = Y(A_y, M) = \{(y,m):\ y \in A_y,\ m \in M\}$, $A_y \in [0,1]$. The averaged form $W(A_w, M) = Av(V)$ of a Boolean array can also be normalized resulting in $\mathrm{Norm}(W) = Z(A_z, M) = \{(z,m):\ z \in A_z,\ m \in M\}$, where $A_z = \{0, 1/q, 2/q, \ldots, 1\}$, the normalized averaged state being computed as follows.

$$z(m) = \frac{1}{q} \sum_{k=0}^{q} v(\phi_k(m)). \tag{5}$$

From (5) it is seen that $z(m) = 1$ can take place when all the cells in the averaging area of $V(B, M)$ have states equal to "1".

Hence follows that a Boolean array represents a spatial function through the distribution of "ones" over a discrete space. Averaging is the procedure of computing the density of this distribution, which transfers a Boolean array into a cellular array with real state values from a discrete alphabet. The inverse procedure of obtaining a Boolean array representation of a given cellular array with real state values is more important and more complicated.

A Boolean array $V(B, M)$ which averaged form $W(A_w, M) = Av(V)$ approximates a given cellular array $U(\boldsymbol{R}, M)$ is called its *Boolean discretization.* Consequently, $V(B, M)$ is a Boolean discretization of $Y(A_y, M) = \mathrm{Norm}(U)$ if $\mathrm{Norm}(Av(V)) = Z(A_z, M)$ approximates $Y(A_y, M)$. The approximation error should be limited by a certain value, which is more convenient to express through normalized values, i.e.,

$$z(m) - y(m) \leq \epsilon \quad \text{for any} \quad m \in M, \tag{6}$$

where ϵ is an *admissible approximation error.*

3. Boolean discretization method

The problem of Boolean discretization is stated as follows. Given a normalized cellular array $Y(A_y, M)$, and an admissible approximation error ϵ, a Boolean array $V(A_v, M) = \mathrm{Disc}(Y)$ should be obtained, such that the condition (6) be satisfied. The problem solution is based on the fact, that for any $m \in M$ the probability of the event $v(m) = 1$ is equal to $y(m)$, i.e.,

$$P_{(v(m)=1)} = y(m). \tag{7}$$

The above simple rule follows directly from the probability definition, provided $y(m)$ is assumed to be constant on the averaging area. The rule results in a Boolean array, where the expected value $\mu(y(m))$ is equal to the mean state $y'(m)$ over the averaging area $Av(m)$, i.e.,

$$\mu(y(m)) = \frac{1}{q} \sum_{k=0}^{q} v(\phi_k(m)) P_{(v(\phi_k(m))=1)} = \frac{1}{q} \sum_{k=0}^{q} y(\phi_k(m)) = y'(m). \tag{8}$$

From (8) follows that the approximation error ϵ vanishes in those cells, where

$$y(m) = y'(m) = \frac{1}{q} \sum_{k=0}^{q} y(\phi_k(m)). \tag{9}$$

A set of such cases includes, for example, all linear functions and parabolas of odd degrees, considered on the averaging area relative to a coordinate system with the origin in the cell named m. When (9) is not satisfied, the error of Boolean discretization $e(m) = z(m) - y(m) \neq 0$ is the largest in the cells where $y(m)$ has extremes.

A discretization algorithm which pretends to be accurate can be also obtained. Its main part should comprise the construction of a cellular array $\Upsilon(A_y, M) = \{(\gamma, m)\}$ such that for all its cells the mean value over the averaging area be equal to the corresponding cell state of the initial cellular array $Y(A_y, M)$, i.e.,

$$y(m) = \frac{1}{q} \sum_{k=0}^{q} \gamma(\phi_k(m).$$

This method requires great computational costs, because a determination of each cell state in $\Upsilon(A_y, M)$ includes the solution to a system of $|Av(m)|$ equations. Since these computations should be made in the floating-point format, the complete accuracy may not be ensured. So, we consider this method impractical and further propose some practical techniques to increase the accuracy of the above approximate simple rule (7).

From (8) it is clear that the error $e(m)$ depends on the function behavior in $Av(m)$ and the cardinality $q = |Av(m)|$, these two parameters being conditioned by a spatial discretization step h. The latter should be small, allowing q to be chosen large enough to smooth function extremes.

There is one more parameter which determines the accuracy of Boolean discretization. It is the *error dispersion* σ, which characterizes the coarseness of discrete function representation. In the cellular automata theory it is called *automata noise* and considered as an important parameter, closely related to computation errors. Hence, it is better to express the discretization accuracy by the two values:

$$E = \frac{1}{M} \sum_{m \in M} e(m), \qquad \sigma = \left(\frac{1}{M} \sum_{m \in M} (e(m) - E)^2 \right)^{1/2}. \tag{10}$$

So, the accuracy requirements should be given as a pair of inequalities,

$$E < \epsilon, \qquad \sigma < \eta, \tag{11}$$

where ϵ and η are admissible values of the approximation error and the error dispersion.

The above two requirements may be contradictory. This occurs, when the function has sharp extremes. In this case the dispersion minimization

requires large $q = |Av(m)|$, while for the mean error decrease it should be taken small enough. The latter is due to the fact that for functions which are either convex or concave on a certain spatial interval, the difference between the mean and the extreme values increases with an increase of the interval. The only way to satisfy both requirements is to increase the cardinality of a naming set. There is two ways of doing this, shown in the next section.

4. Providing Boolean discretization accuracy

The first way to provide Boolean discretization accuracy is to take a naming set of large cardinality, allowing the averaging area size be also chosen large enough. The following example shows how the accuracy parameters depend upon the averaging area size.

Example 1. Boolean discretization of a half of one-dimensional wave $y = \sin x$ with $0 < x < \pi$ is to be obtained in order that an experimental assessment of the discretization accuracy be performed. Taking $h = \pi/N$ for the space discretization, the normalized cellular array

$$y(i) = a \sin \frac{\pi i}{N}, \quad i = 0, 1, \ldots, N,$$

is found. Two Boolean discretizations $V_1(B, M_1)$ and $V_2(B, M_2)$, with $M_1 = \{0, 1, \ldots, 179\}$ and $M_2 = \{0, 1, \ldots, 359\}$, of $Y(A_y, M)$ are obtained according to (7). For both of them, a number of averaged cellular arrays with different averaging area size have been obtained, and the mean error together with the dispersion have been calculated for each case (Table 1).

Table 1. Dispersion σ, main error E, and the error $e(N/2)$ experimentally obtained for Boolean discretization of $y = \sin x$ with different averaging area size

q	$N = 180$			$N = 360$		
	σ	E	$e(90)$	σ	E	$e(180)$
15	0.11	0.070	0.000	0.011	0.075	0.000
29	0.066	0.070	0.000	0.005	0.057	0.000
51	0.031	0.034	−0.019	0.002	0.047	0.000
61	0.030	0.030	−0.049	0.001	0.041	0.000
101	0.033	0.089	−0.079	0.001	0.025	−0.001
105	0.041	0.099	−0.0086	0.001	0.024	−0.095
201				0.021	0.073	−0.0089

From Table 1 the following is clearly seen:

1. Both accuracy parameters: dispersion and mean error are the smaller the larger is the naming set cardinality, i.e., the smaller is the spatial step. It should be mentioned, that a correct comparison is the one made between the

Boolean arrays having equal averaging area size, and those measured in the initial continuous space, i.e., having equal hq. In our case, compared should be pairs of the cases for which $q_{N=180} = 2q_{N=360}$. For example, the case $q_{N=180} = 51$ should be compared to the one with $q_{N=360} = 101$, resulting in the fact that both accuracy parameters with $N = 360$ are about 5 times better than those with $N = 180$.

2. The mean error and the dispersion considered versus the averaging area size have minimum values with $q_{N=180} = 61$ and with $q_{N=360} = 105$, which corresponds to $q_{\min} \approx N/3$.

3. With the growth of the averaging area the error on the function extreme increases. In our example it is shown by the error $e(N/2)/q$.

The second method of improving Boolean discretization parameters is to take the naming set of a Boolean discretization one dimension larger than that of the initial cellular array. Hence, in order that Boolean discretization of a cellular array $Y(A_y, M)$ be obtained Boolean array $V(B, M \times L)$ is to be constructed, whose naming set is a multilayered structure

$$M = \bigcup_{l=1}^{L} M_l, \quad M_l = \{m_1^{(l)}, \ldots, m_N^{(l)}\}.$$

The state values $y(m_i^{(l)})$ are obtained in all the layers in one and the same way according to the rule (7). The averaging area determined according to (3) in each layer forms a multilayer subarray of the total size $Q = q \times L$. The averaged array $W(A_w, M)$ is again a one-layer array, where the cell states are computed as follows:

$$w(m) = \sum_{l=1}^{L} \sum_{k=0}^{q} v(\phi_k(m^{(l)}) \quad \forall m_i^{(l)} \in M, \tag{12}$$

the normalized averaged state being $z(m) = w(m)/Q$.

Example 2. The above method of increasing Boolean discretization accuracy has been experimentally verified as follows. For the function $y = \sin x$ from Example 1 Boolean discretization $V(B, M \times L)$ with $|M| = N = 360$, $L = 10$ have been obtained according to (7), as applied to all the layers cells. The averaging of $V(B, M \times L)$ has been performed with different values of q. The results presented in Table 2 show that better accuracy parameters are achieved than those in a one-layer array with the same N. The price is a tenfold increase of the Boolean array size.

One more error elimination method called *extreme compensation method* is further proposed, which is useful when a cellular array $Y(A_y, M)$ under Boolean discretization has sharp extremes or breaks of the initial function

$q \times L$	σ	E	$e(180)$
5×10	0.064	0.048	0.000
15×10	0.028	0.031	0.000
29×10	0.022	0.026	0.000
51×10	0.017	0.0122	0.000
61×10	0.015	0.019	-0.003
101×10	0.008	0.008	-0.014

Table 2. Dispersion σ, the main error E, and the error $e(N/2)$ experimentally obtained for a number of multilayered Boolean discretizations of $y = \sin x$

$y(m)$. The cells, where the function has the above peculiarities, are further referred to as *extremal cells* denoted as m^*. The method provides for replacing each subarray $Av(m^*)$ by another one, $Av^*(m^*)$, called the *virtual averaging area.* After such a replacement a new *virtual* cellular array is obtained, i.e.,

$$Y^*(A_y, M) = Y(A_y, M) \setminus \sum_{m_i^*=1}^{s} Av(m_i^*) \bigcup \sum_{m_i^*=1}^{s} Av^*(m_i^*), \qquad (13)$$

where $\{m_1^*, \ldots, m_i^*, \ldots, m_s^*\}$ is a set of extremal cells names.

Each set $Av^*(m_i^*)$ differs from $Av(m_i^*)$ in the cell states, whose values $y^*(\phi_k(m^*)$ should satisfy the condition,

$$z(m) - y(m) < \epsilon' \quad \forall (z, m) \in Av(\mathrm{Disc}(Y^*(A_y, M))), \qquad (14)$$

where $\epsilon' < \epsilon$. Condition (14) means that after the discretization of $Y^*(A_y, M)$ and averaging the result the array $Z(A_z, M)$ is obtained, approximating $Y(A_Y, M)$ with a very small error. To obtain these values each $y(\phi_k(m^*)) \in Av(m^*)$ should be augmented by a value $\tilde{y}(\phi_k(m^*))$, symmetric to $y(m)$ relative to the constant function $c(m) = y(m^*)\ \forall\, m \in Av(m^*)$, i.e.,

$$\tilde{y}(\phi_k(m^*)) = 2y(m^*) - y(\phi_k(m^*)) \qquad (15)$$

with $\phi_0(m) = \phi_0(m^*) = m^*$. The augmentation of the state values in the virtual averaging areas are done as follows

$$y^*(\phi_k(m^*) = \frac{1}{2}\Big(y(\phi_k(m^*) + \tilde{y}(\phi_k(m^*))\Big) = y(m^*). \qquad (16)$$

From (16) it is easily seen, that when the function under Boolean discretization is piecewise linear, all the cell states in $Av^*(m^*)$ are equal to $y(m^*)$, i.e.,

$$Av^*(m^*) = \{(y(m^*), \phi_k(m^*)) :\ k = 0, \ldots, q\}. \qquad (17)$$

So, in many cases it makes sense to obtain a piecewise linear approximation of $Y(A_y, M)$, and then perform Boolean discretization with the use of

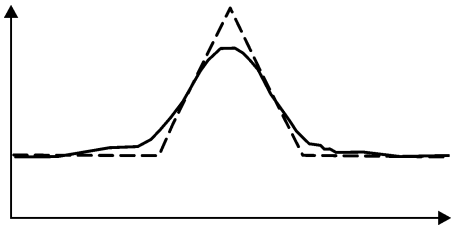

Figure 1. Boolean discretization of a piecewise linear function: the initial function $y(m)$ (dashed line) and the averaged Boolean $z(m)$ (solid line) obtained by applying the rule (7) to $y(m)$

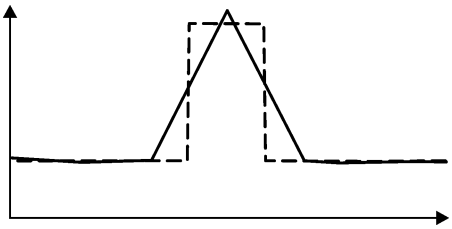

Figure 2. Boolean discretization of a piecewise linear function: the virtual function $y^*(m)$ (dashed line) and the averaged Boolean $z(m)$ (solid line) obtained by the extreme compensation method

the extreme compensation method. Of course, a spatial discretization step should be chosen in such a way that the distance between the two nearest extremes be larger than $2r$, r being a radius of the averaging area. The efficiency of the method is illustrated by the results of Boolean discretization of a piecewise linear function $Y(m)$, shown in Figures 1 and 2. From them, it is clearly seen that the use of the extreme compensation method is much more accurate.

5. Cellular array arithmetic

Simulation of the complex spatial dynamics with cellular automata requires the perform once of a few operations on Boolean arrays. The most important are the following: 1) addition (subtraction) of two Boolean arrays, 2) addition (subtraction) of a real cellular array to a Boolean one, 3) multiplication of two Boolean arrays, and 4) multiplication of a Boolean array by a real cellular array. All these operations are determined in the domain and range of a set of cellular arrays, belonging to one and the same class $K(M,T)$, characterized by a naming set M, and a set of the naming functions $T = \{(\phi_k(m)), k = 0,\dots,q\}$ in $Av(m)$. Particularly, a Boolean array $V(B,M')$ with $Av'(m') = \{(v,\phi'_k(m')),\ v = 0,1,\ k = 0,\dots,q\}$ belongs to the class $K(M,T)$ if

1) $M' = M$,
2) In $Av'(m')$, the set $\{\phi'_k(m') :\ k = 0,\dots,q\} = T$.

Boolean arrays addition (subtraction). A Boolean array $V(B,M)$ is called a sum of $V_1(B,M)$ and $V_2(B,M)$, which is written down as

$$V(B,M) = V_1(B,M) \oplus V_2(B,M) \tag{18}$$

if its averaged normalized form $Z(A_z, M) = \mathrm{Norm}(Av(V))$ is a matrix-like sum of $Z_1(A_z, M) = \mathrm{Norm}(Av(Z_1))$ and $Z_2(A_z, M) = \mathrm{Norm}(Av(Z_2))$.

This means that for any $m \in M$ $z(m) = z_1(m) + z_2(m)$, where (z, m), $z_1(m)$, $z_2(m)$ are the cell states in $Z(A_z, M)$, $Z_1(A_z, M)$, $Z_2(A_z, M)$, respectively, or, according to (5)

$$\sum_{k=0}^{q} v(\phi_k(m)) = \sum_{k=0}^{q} v_1(\phi_k(m)) + \sum_{k=0}^{q} v_2(\phi_k(m)). \tag{19}$$

Using (7), the resulting array can be obtained by allocating the "ones" in the cells of an empty array $\Omega = \{(0, m) : \forall\, m \in M\}$ with the probability

$$P_{(0\to 1)} = \frac{1}{q}\left(\sum_{k=0}^{q} v_1(\phi_k(m)) + \sum_{k=0}^{q} v_2(\phi_k(m))\right) = z_1(m) + z_2(m). \tag{20}$$

When the Boolean array addition is used as an intermediate operation, it is more convenient to obtain the resulting array by means of updating one of the operands so that it be equal to the resulting Boolean array. This can be done as follows. Let $V_1(B, M)$ be changed into $V_1(B, M) \oplus V_2(B, M)$. Then some cells $(v_1, m) \in V_1(B, M)$, out of those, where $v_1(m) = 0$, should invert their states. Hence, the probability of such an inversion should be the relation of the value to be added to the amount of "zeros" in the averaging area $Av(m) \in V_1(B, M)$.

$$P_{(0\to 1)} = \frac{z_2(m)}{(1 - z_1(m))}. \tag{21}$$

Subtraction can also be performed in two ways. The first is similar to (20), the resulting difference $V(B, M) = V_1(B, M) \ominus V_2(B, M)$, being obtained by allocating the "ones" in the cells of an empty array with the probability

$$P_{(0\to 1)} = z_1(m) - z_2(m). \tag{22}$$

The second is similar to (21), bearing in mind that the inversion should be done in the cells with the states $v_1(m) = 1$, the probability of the inversion being computed as follows:

$$P_{(1\to 0)} = \frac{z_2}{z_1}. \tag{23}$$

Addition (subtraction) of real and Boolean cellular arrays. A Boolean array $V(B, M)$ is the sum of a Boolean array $V_1(B, M)$ and a normalized cellular array $Z_2(A_z, M)$ if in $Z(A_z, M) = \mathrm{Norm}(Av(V))$ each cell state $z(m) = z_1(m) + z_2(m)$, $z_1(m)$, $z_2(m)$ being the states of the cells in $Z_1(A_z, M) = \mathrm{Norm}(Av(V_1))$ and $Z_2(A_z, M)$, respectively, i.e.,

$$\sum_{k=0}^{q} v(\phi_k(m)) = \sum_{k=0}^{q} v_1(\phi_k(m)) + z_2(m). \tag{24}$$

The resulting array can be obtained by allocating the "ones" in the cells of an empty array with the probability

$$P_{(0\to 1)} = \frac{1}{q}\sum_{k=0}^{q} v_1(\phi_k(m)) + z_2(m). \tag{25}$$

When such an addition is used as an intermediate operation, which is the case in the hybrid method of the diffusion-reaction simulation [5], the result is obtained by the inverting states $v_1(m) = 0$ into $v_1(m) = 1$ like in (21) with the probability

$$P_{(0\to 1)} = \frac{z_2(m)}{1 - \frac{1}{q}\sum_{k=1}^{q} v_1(\phi(m))}. \tag{26}$$

Similarly, the subtraction $V(B,M) = V_1(B,M) \ominus Z_2(B,M)$ is performed according to one of the two following formulas: either to allocate the "ones" in an empty array with the probability

$$P_{(1\to 0)} = \frac{1}{q}\sum_{k=1}^{q} v_1(\phi(m)) - z_2(m), \tag{27}$$

or to invert the cell states $v_1(m) = 1$ in $V_1(B,M)$ with the probability

$$P_{(1\to 0)} = \frac{z_2}{\frac{1}{q}\sum_{k=1}^{q} v_1(\phi(m))}. \tag{28}$$

Multiplication of two Boolean arrays. A Boolean array $V(B,M)$ is called the product of $V_1(B,M)$ and $V_2(B,M)$, which is written down as:

$$V(B,M) = V_1(B,M) \otimes V_2(B,M) \tag{29}$$

if its averaged normalized form $Z(A_z,M) = \mathrm{Norm}(Av(V))$ has cell states, which are the products of the corresponding cell states $z_1(m)$ of $Z_1(A_z,M) = \mathrm{Norm}(Av(V_1)))$ and $z_2(m))$ of $Z_2(A_z,M) = \mathrm{Norm}(Av(V_2)$. This means that

$$\frac{1}{q}\sum_{k=0}^{q} v(\phi_k(m)) = \frac{1}{q}\sum_{k=0}^{q} v_1(\phi_k(m)) \times \frac{1}{q}\sum_{k=0}^{q} v_2(\phi_k(m)) \tag{30}$$

The resulting array may be obtained by allocating the "ones" in the cells of an empty array with the probability

$$P_{(0\to 1)} = \frac{1}{q}\sum_{k=0}^{q} v_1(\phi_k(m)) \times \frac{1}{q}\sum_{k=0}^{q} v_2(\phi_k(m). \tag{31}$$

Multiplication of a Boolean array by a real cellular array. A Boolean array $V(B,M)$ is the product of a Boolean array $V_1(B,M)$ and a normalized cellular array $Z_2(A_z,M)$, which is written down as:

$$V(B,M) = V_1(B,M) \otimes Z_2(A_z,M), \tag{32}$$

if its averaged normalized form $Z(A_z,M) = \mathrm{Norm}(Av(V))$ has the cell states, which are the products of corresponding cell states $z_1(m)$ of $Z_1(A_z,M) = \mathrm{Norm}(Av(V_1))$ and $z_2(m)$ of $Z_2(A_z,M)$, i.e.,

$$\frac{1}{q}\sum_{k=0}^{q} v(\phi_k(m)) = \frac{z_2(m)}{q}\sum_{k=0}^{q} v_1(\phi_k(m)). \tag{33}$$

The resulting arrays are obtained by allocating the "ones" in the cells of an empty array with the probability

$$P_{(0\to 1)} = \frac{z_2(m)}{q}\sum_{k=0}^{q} v_1(\phi_k(m)), \tag{34}$$

Clearly, the multiplication of a Boolean array $V_1(B,M)$ by a constant $a \in A_z$, $A_z = \{0, 1/q, \ldots, 1\}$ is the same as multiplication $V(B,M)$ by $Z_2(a,M)$ with all the cells having the equal states $z_2(m) = a$.

6. Conclusion

Methods of transforming spatial continuous functions into their representation in the form of Boolean arrays are proposed. Such transformations, as well as the formal arithmetic operations in the domain of cellular arrays, are required when cellular automata models are used to simulate the spatial dynamics. This is especially important when certain components of the process under simulation are modelled by cellular automaton evolution and others are given in the real domain (for example, the diffusion-reaction processes). As such discrete-continuous transformations are approximate, some techniques for providing admissible approximation accuracy are also proposed and investigated.

References

[1] Wolfram S. Theory and Applications of Cellular Automata. – Singapore: World Scientific, 1986.

[2] Bandman O. Fine-grained parallelism in mathematical physics // Programmirovanie. – 2001. – № 4. – P. 5–20.

[3] Wolfram S. New Kind of Science. – Champain, USA: Wolfram Media, Inc., 2002.

[4] Toffolli T. Cellular automata as an alternative to (rather than approximation) to differential equations in modelling physics // Physica. – 1984. – Vol. 10 D. – P. 117–127.

[5] Bandman O. Cellular-neural automaton: a hybrid model for reaction-diffusion simulation // Future Generation Computer Systems. – 2002. – Vol. 18. – P. 737–745.

[6] Rothman D.H., Zaleski S. Lattice-Gas Cellular Automata. Simple Models of Complex Hydrodynamics. – Cambridge University Press, 1997.

[7] Chua L. CNN: A Paradigm for Complexity. – Singapore: World Scientific, 1999.