Cellular Automata Composition Techniques for Spatial Dynamics Simulation

Olga Bandman

Supercomputer Software Department, ICM&MG, Siberian Branch of Russian Academy of Sciences, Novosibirsk, 630090, Russia, bandman@ssd.sscc.ru

1 Introduction

A Cellular Automaton (CA) is nowadays an object of growing interest as a mathematical model for spatial dynamics simulation. Due to its ability to simulate nonlinear and discontinuous processes, CA is expected [1, 2] to become a complement to partial differential equations (PDE). Particularly, CA may be helpful when there is no other mathematical model of a phenomenon which is to be investigated. By now, a great variety of CA are known, whose evolution simulates certain kinds of spatial dynamics. All of them are descendants of a classical von-Neumann's CA, which has a Boolean alphabet, deterministic single-cell updating transition functions, and a synchronous mode of operation. Such CA are capable to simulate a number of processes in physics, chemistry, biology, as well as a behavior of colonies of animals or crowds of peoples. The most known are CA-models of physical processes, such as diffusion [1, 3], wave propagation [4], phase transition [2, 5], spatial self-organization [6], etc. More complicated CA called Gas–Lattice models [7] are used in hydrodynamics, some of them [8, 9] dealing with a real alphabet. Among the above CA-models there are those, which may be described in terms of PDE. Such are diffusion, liquid flow, solitons. The first two are proved to correspond strictly to their PDE counterparts [10, 11]. As for soliton [4] - its CA-model is a bright manifestation of a huge difference in complexity between a third order nonlinear PDE and a simple Boolean function that simulate the same phenomenon.

In chemistry and microelectronics [14] asynchronous probabilistic CA–models with multi-adjustable cells (in chemistry being referred to as Monte-Carlo methods), are intensively used for studying surface reaction on catalysts [12, 13] and processes of epitaxial growth of crystals [14]. The processes are simulated by mimicking real movements and interactions of atoms and molecules. This type of processes have no mathematical description in term of PDE, because continuous functions cannot capture the behavior of discrete particles.

Biology and medicine present a wide field of processes to be simulated by CA–models, genetics [15], myxobacteria swarming [16], growth of tumor [17] being the examples. In solving ecological problems, CA are used more and more frequently to simulate the propagation of diseases [18] and of fire in the forests, evolution of populations, etc. Recently, CA–models have aroused considerable interest in simulating the crowds behavior [19, 20]. Moreover, CA–simulation has now gone beyond the scope of scientific research, being used, for example, to simulate the process of cement hardening [21].

The diversity of processes being simulated by CA caused the necessity to extend the cellular automaton concept by allowing it to have any kind of alphabet (Boolean, integer, real, symbolic), any kind of transition functions (deterministic, probabilistic), and any mode of functioning (synchronous, asynchronous). Although, the imperative properties of classical CA still remain. They are as follows.

1) CA consists of many identical simple processing units (cells).

2) Interactions between cells are constrained by a small (relatively to the total amount of cells) neighborhood.

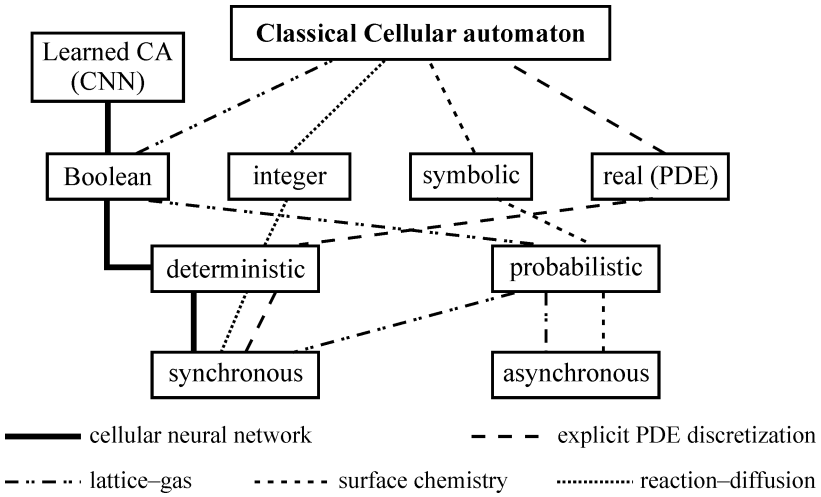


Fig. 1. Properties of CA simulation models: lines connecting the rectangles show the sets of properties, characterizing certain types of CA–models

Such an extended concept of CA is sometimes referred to as *fine-grained parallelism* [22]. A wide class of CA exhibiting the properties of self-organization and emergency are considered also as models of *complex systems* [23]. Nevertheless, hereafter the term CA or CA-model is used as the most habitual one. In Fig.1, CA–model types are collected and allocated according to their properties. It is worth noting that Cellular Neural Networks (CNN) [24] and explicit form of discrete representation of Partial Differential Equations (PDE)

are also regarded as special cases of CA–models because two above properties are inherent in them.

Unfortunately, there is no formal procedure to construct a CA–model according to a given qualitative or quantitative specification of a space–time process. All known models are the result of a trial and error work based on high level of experience in CA modeling, and sophisticated understanding of the phenomenon to be simulated. However now, having a relatively large bank of CA–models at hand, the problem arises how to construct CA–models of complicated real life processes, which may be thought of as a set of interacting simple ones, for whom CA-models are known. Hence, methods and tools are needed to organize the interaction of CA in such a way that the evolution of a composed CA represents the required spatial process. The problem is similar to that in mathematical physics where a PDE is represented as the sum of differential operators of certain degrees, each term having its own physical meaning.

Complications in CA composition techniques are associated with the Boolean alphabet, because the overall impact of a Boolean CA component may not be obtained by means of conventional arithmetics. Even more difficult is to compose CA–models having different alphabets and/or different modes of operation which is the case in reaction–diffusion and prey–predatory processes, where diffusion is given as a Boolean CA, and reaction – as a real function. For example, the snowflakes formation is usually simulated by a Boolean CA, while if it proceeds in active medium, a chemical component should be added, which may be given as a nonlinear real function. The first prototype of such a composition is proposed in [26] for combining a nonlinear reaction function with a Boolean diffusion, and a more general probabilistic variant is given in [27].

From the above it follows that the CA composition methods should be provided with appropriate techniques for performing algebraic operations on CA configurations with all kind of admissible numerical alphabets: Boolean, real and integer. This means that some kind of equivalent transformations should be introduced to make CA–models compatible with different alphabets. Based on such a transformations a special algebra on CA configurations is constructed, which allows us to combine the functioning of several CA–models in a single complex process [28].

A fast increase of the variety of CA-models and the growing necessity of simulating complicated processes require a general formal approach to CA composition, which is to be valid for any type of CA and any type of their interaction. It is precisely the object of the Chapter, which aims to present a theoretical foundation, and based on it, the CA composition techniques in a generalized and systematic form. To capture all features of essential diversity of CA–models, the more general formalism for CA–algorithms representation, namely, Parallel Substitution Algorithm (PSA) [25], is chosen as a mathematical tool.

The Chapter combines the results on the subject that are scattered about the papers. It consists of the following sections. In the next section, main concepts and formal definitions are given and operations on cellular arrays are defined. Third section presents a sequential composition of CA–models. In the fourth section a parallel and a mixed composition methods are given. The fifth section is concerned in computational properties of composed CA, namely, accuracy, stability and complexity. All composition methods are illustrated by the original simulation results.

2 Main Concepts and Formal Problem Statement

For simulation of spatial dynamics, an extended concept of CA–model is further considered, whose expressive power is sufficient for simulating natural phenomena of several kinds. The concept is based on the PSA formalism [25], which, though intended for parallel hardware design, seems to be the most suitable for modeling composite processes. Moreover, due to its flexibility, PSA allows the strict formulation of most important requirements imposed on CA composition techniques: 1) conservation of behavioral correctness, and 2) alphabets compatibility of several CA involved in the composition.

2.1 Formal Definition of a CA–model

Simulation of a natural phenomenon comprises the determination of a suitable mathematical model, the development of an appropriate algorithm and a computer program, and using the latter for computing desirable functions of time and space. If CA is chosen as a mathematical model, then time is a discrete sequence of nonnegative integers, space is a discrete set referred to as a *naming set*, function values are from an appropriate *alphabet*.

A finite naming set $M = \{m_k : k = 0, \ldots, |M|\}$ is further taken for the space. Its elements $m_k \in M$ in simulation tasks are usually represented by the integer vectors of coordinates of a Cartesian space of finite size. For example, in 2D case $M = \{(i, j) : i = 0, 1, \ldots, I,\ j = 0, 1, \ldots, J\}$. A notation m is used instead of (i, j) for making the general expressions shorter and for indicating, that it is valid for any other kind of discrete space points.

No constraint is imposed on the alphabet A. The following cases are further used: $A_S = \{a, b, ..., n\}$ - a finite set of symbols, $A_B = \{0, 1\}$ — the Boolean alphabet, $A_R = [0, 1]$ — a set of real numbers in a closed continuous interval. Symbols from the second part of the Latin alphabet $\{v, u, x, y, \ldots, z\}$ are used to denote the variables defined on A. Appealing to the above extended concept of alphabet is dictated by the aim of the study – to combine several CA of different types into a single one for simulating composite phenomena. A pair (a, m) is called a *cell*, $a \in A$ being a cell–state and $m \in M$ — a cell–name. To indicate the state of a cell named by m both notations $u(m)$ and u_m are further used.

The set of cells

$$\Omega = \{(u, m) : u \in A, m \in M\}, \tag{1}$$

such that there are no cells with identical names, is called a *cellular array*, or, sometimes, a *global configuration* of a CA.

On the naming set M, a mapping $\phi : M \to M$ is defined, referred to as a *naming function*. It determines a *neighboring cell* location $\phi(m)$ of a cell named m. In the naming set of Cartesian coordinates $M = \{(i, j)\}$, the naming functions are usually given in the form of shifts $\phi_k = (i + a, j + b)$, a, b being integers. The set of naming functions determines a *template*.

$$T(m) = \{\phi_0(m), \phi_1(m), \ldots, \phi_n(m)\}, \tag{2}$$

which associates a number of cell names to each name $m \in M$. The cell named as $\phi_0(m)$ is called an *active cell* of a template, where $n \ll |M|$, and $\phi_0(m) = m$ by condition.

A subset of cells

$$S(m) = \{(u_0, m), (u_1, \phi_1(m)), \ldots, (u_n, \phi_n(m))\}, \tag{3}$$

with the names from $T(m)$ is called a *local configuration*, $T(m)$ being its *underlying template* . The set

$$U_S(m) = (u_0, u_1, \ldots, u_n)$$

forms a *local configuration state vector* .

A cell (u_k, m) changes its state u_k to the next–state u'_k under the action of a *local operator*, which is expressed in the form of a *substitution* [25] as follows

$$\theta(m) : S(m) \star S''(m) \to S'(m), \tag{4}$$

where

$$\begin{aligned} S(m) &= \{(v_0, m), (v_1, \phi_1(m)), \ldots, (v_n, \phi_n(m))\} \\ S'(m) &= \{(u'_0, m), (u'_1, \phi_1(m)), \ldots, (u'_n, \phi_n(m))\} \\ S''(m) &= \{(v_{n+1}, \phi_{n+1}(m), \ldots, (v_{n+h}, \phi_{n+h}(m))\}. \end{aligned} \tag{5}$$

In (5) $S(m)$, $S'(m)$ and $S''(m)$ are local configurations, the first two having the same underlying template, and the third one comprises h additional cells, which together with $S(m)$ conditions the next state values u'_k.

The next-states u'_k, $k = 0, 1, \ldots, n$, of the cells from S' are values of the *transition functions* f_k of the cell states from $S(m) \cup S''(m)$, i.e.

$$u'_k = f_k(v_0, \ldots, v_n, \ldots, v_{n+h}), \quad \forall k = 0, 1, \ldots n. \tag{6}$$

A union of the left-hand side local configurations $S(m) \cup S''(m)$ in (4) is called a *a cell–neighborhood* where S'' is a *a context*, $S(m)$ is *a base* of $\theta(m)$. The right-hand side $S'(m)$ is the *next-state base* of the local operator.

The underlying templates $T(m)$, $T'(m)$, and $T''(m)$ of the local configuration in (5) are in the following relation.

$$T'(m) = T(m),$$
$$T(m) \cap T''(m) = \emptyset, \tag{7}$$

$T(m)$ being referred to as a *basic template* of θ.

A local operator $\theta(m)$ is said to be applicable to a cell named $m \in M$ if $S(m) \cup S''(m) \subseteq \Omega$. Otherwise, it is not applicable. Application of $\theta(m)$ to a certain cell (v, m) (*a single-shot application*) means execution of the following actions. For all $k = 0, \ldots, n$

1) the next-states u'_k are computed according to (6),

2) the cells $(v_k, \phi_k(m)) \in S(m)$ are updated by replacing the cell states u_k by u'_k.

The cells $(v_{n+l}, \phi_{n+l}(m))$, $l = 0, \ldots, h$, from the context remain unchanged. They play a role of an application condition, the states being used as variables in the transition functions (Fig.2).

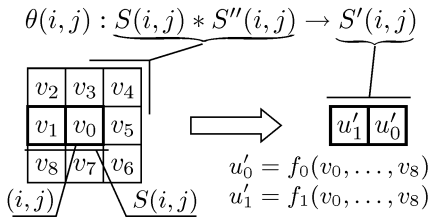


Fig. 2. Graphical representation of a local operator

A subset $\hat{M} \subseteq M$, referred to as the *active naming set* is defined, such that it comprises the names of active cells, i.e., the cells to which the local operator is applied to make the whole cellular array transit to the next state. Application of θ to all active cells $m \in \hat{M}$ comprises *an iteration* performing a *global transition.*

$$\Phi(\hat{M}) : \Omega(t) \rightarrow \Omega(t+1), \tag{8}$$

A sequence of global transition results

$$\Sigma(\Omega) = (\Omega, \Omega(1), \ldots, \Omega(t), \Omega(t+1), \ldots, \Omega(\hat{t})) \tag{9}$$

is called a *CA evolution.*

The CA evolution is the result of a simulation task, representing the process under simulation. If the process converges to a stable global state, then CA evolution has a termination, i.e., there exists such a $t = \hat{t}$, that

$$\Omega(\hat{t}) = \Omega(\hat{t}+1) = \Omega(\hat{t}+2) = \ldots = \Omega(\hat{t}+\xi), \tag{10}$$

where ξ is an a priori given number. If it not so, then the evolution is infinite, i.e., exhibits an oscillatory or chaotic behavior [2].

There are different modes of ordering local operator application in space and time to perform a global transition from $\Omega(t)$ to $\Omega(t+1)$. The following are the most important ones.

Synchronous mode provides for transition functions (6) being computed using the current state values of all their variables, i.e.

$$S(m) \cup S''(m) \in \Omega(t). \tag{11}$$

The transition to the next cell–state values occurs after all the transition functions in cells from $S(m)$ for all $m \in \hat{M}$ are computed. Theoretically, it may be done in all cells simultaneously or in any order, which manifests the *cellular parallelism*. In fact, when a conventional sequential computer is used, such a cellular parallelism is imitated by delaying cell updating until all next states are obtained. So, the cellular parallelism is a *virtual parallelism*, which cannot be for the benefit when CA–model is run on conventional computers.

Asynchronous mode of operation suggests no simultaneous operations (neither real nor virtual). Intrinsic parallelism of CA–models is exhibited by the arbitrary order of cells to be chosen for application of $\theta(m)$, the updating of cell states of $S'(m)$ being done immediately after $\theta(m)$ is applied. The time of such an application is referred to as a *time-step* and denoted as τ. So, each global transition $\Omega(t) \to \Omega(t+1)$ consists of $|\hat{M}|$ sequential time steps, forming a sequence of cellular arrays

$$\gamma_\alpha(\Omega(t)) = \Omega(t), \Omega(t+\tau), \ldots, \Omega(t+|\hat{M}|\tau), \tag{12}$$

which is referred to as *global state transition sequence*. The important property of asynchronous mode of operation is that the state values used by transition functions (4) may belong both to $\Omega(t)$ and to $\Omega(t+1)$, i.e.,

$$S(m) \cup S''(m) \subset \Omega(t) \cup \Omega(t+1). \tag{13}$$

It is the reason why two CA-models with equal $\langle A, M, \hat{M}, \theta \rangle$ starting from the same Ω may have quite different evolutions when operating in different modes. Although, some exotic "very good" CA–models are known, whose evolutions and attractors are invariant whatever mode of operation is used [25].

Multi-stage synchronous mode is also frequently used. It is a mixed mode of operation, which may be regarded both as a synchronized asynchronous mode, and as an asynchronized synchronous one. The mode suggests the whole cellular array of the CA to be partitioned into nonintersecting *blocks* each containing b cells. The block partition induces a dual of partition $\{M'_1, \ldots, M'_b\}$, whose subsets are further called *stage naming subsets* . They contain representative names of all blocks (one out of each block), so that $\hat{M} = M'_1 \cup \ldots \cup M'_b$. Respectively, the iteration is divided into b stages. At each kth stage the local

operator is applied to the cells of M_k synchronously, the stages being processed in asynchronous manner. Naturally, cellular parallelism is here limited by the subset cardinality.

No matter what is the mode of operation, a *global operator* is the result of application of $\theta(m)$ to all cells $m \in \hat{M}$.

From the above it follows that a *CA–model*, denoted as $\aleph$ is identified by five notions:

$$\aleph = \langle A, M, \hat{M}, \theta, \rho \rangle$$

where ρ indicates the mode of operation, $\rho = \sigma$ stands for the synchronous mode, $\rho = \beta$ — for multistage synchronous mode, and $\rho = \alpha$ — for asynchronous mode of local operator application. When the indication of operation mode is essential, the corresponding symbol is placed as an subindex, e.g. $\aleph_\alpha$ denotes an asynchronous CA.

2.2 Correctness of CA Simulation Process

A CA–model $\aleph = \langle A, M, \hat{M}, \theta, \rho \rangle$ is said to be correct (in computational sense) if its operation satisfies the following *correctness conditions*.

1. Non-contradictoriness. *At any moment of time, a cell is allowed to be updated by only one local operator application.* Non-contradictoriness provides absence of conflicts, which are such a situations when a local operator being applied to the cells m and $\phi_k(m)$ simultaneously is attempting to update one and the same cell by writing in it different state values. Formally, non-contradictoriness sufficient condition is formulated as follows [25]: simultaneous application of a local operator to m_k and m_l is allowed only if

$$T'(m_k) \cap T'(m_l) = \emptyset \qquad \forall (m_k, m_l) \in M. \tag{14}$$

It is quite clear, that the non-contradictoriness condition is always satisfied for classical synchronous CA whose local operator has a single–cell base, i.e., $|S'(m)| = 1$. It is not so if $|S'(m)| > 1$, because the local operator has to change several cells simultaneously. For example, a conflict occurs when a surface chemical reaction is simulated where there are reactant molecules occurring in contact (in adjacent cells) producing a pair of other molecules. In the CA–model it corresponds to changing states in both cells simultaneously, which leads to a conflict if synchronous mode is used (Fig.3).

To avoid the above conflict situation, one has to sacrifice a bit of cellular parallelism to non-contradictoriness. It may be done either by constructing an asynchronous CA, simulating the same process, or by replacing the synchronous CA $\aleph_\sigma = \langle A, M, \hat{M}, \theta, \sigma \rangle$ by an equivalent multi–stage CA $\aleph_\beta = \langle A, M, \hat{M}_1, \ldots, \hat{M}_b, \theta, \beta \rangle$. Such a sequalization is done according to the following algorithm.

1. The naming set M is partitioned into $|M|/b$ *blocks*, a block being defined by the underlying template $B(m) = \{\psi_0(m), \psi_1(m), \ldots, \psi_l(m), \ldots, \psi_b(m)\}$ in such a way, that

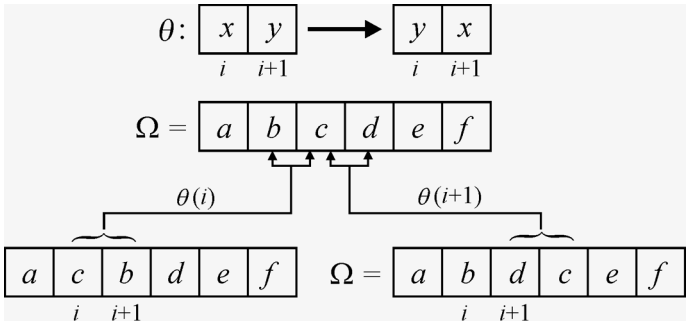


Fig. 3. Graphical representation of contradictoryness property. Simultaneous application of $\theta(i)$ and $\theta(i+1)$ have different results in cells named $i, i+1, i+2$, which engenders a conflict.

$$\begin{aligned} B(m_j) &\supseteq T'(m_j), && \forall j = 1, \ldots, |M|/b. \\ \bigcup_{j=1}^{|M|/b} B(m_j) &= M, && \forall j = 1, \ldots, |M|/b. \\ B(m_h) \bigcap B(m_g) &= \emptyset, && \forall m_h, m_g \in \hat{M}_k, \quad \forall k = 1, \ldots, b, \end{aligned} \tag{15}$$

where $T'(m)$ is the basic template in θ.

2. On the active naming set $\hat{M}$ a stage partition $\{\hat{M}_1, \ldots, \hat{M}_k, \ldots, \hat{M}_b\}$ is defined, i.e.,

$$\hat{M}_k = \{\psi_k(m_j) : \ k = 1, \ldots, b; \ j = 1, \ldots, |M|/b.\} \tag{16}$$

$m_j = \psi_0(m_j)$ being the active cell of a block $B(m_j) \in M$.

3. Each iteration $\Omega(t) \rightarrow \Omega(t+1)$ is divided into b sequential stages $(t_1, t_2, \ldots, t_b)$, $t_b = t+1$, the resulting arrays forming a sequence:

$$\gamma_\beta(t) = \Omega(t), \ldots, \Omega(t+t_k), \Omega(t+t_{k+1}), \ldots, \Omega(t+1), \quad t_k = \frac{\tau k}{b}, \tag{17}$$

referred to as a *stage transition sequence*. On the k-th stage, $k = 1, \ldots, b$, $\theta(m)$ is applied synchronously to all cells from $\hat{M}_k$.

4. The subsets $\hat{M}_k$, $k = 1, \ldots, b$ are processed sequentially in arbitrary order, hence, the total number of possible stage transition sequences is $|\{\gamma_\beta\}| = b!$.

It is quite clear, that a CA–model obtained by the above algorithm satisfies the non-contradictoryness condition (14). Moreover, its evolution although differing from the incorrect initial one, should simulate the wanted process.

As for asynchronous CA, they always satisfy non-contradictoriness conditions, because at each step only one application of $\theta(m)$ is allowed.

Fairness. *At each iteration $\theta(m)$ should be applied to all cells $m \in \hat{M}$, to any cell $m \in \hat{M}$ being applied only once.* Fairness ensures that all cells have equal rights to participate in the CA operation process, therefore, it is sometimes referred to as equality in rights of cells activity [29]. Synchronous classical CA satisfy this property according to the definition of synchronicity. When multi-stage synchronous mode is used, fairness is provided by conditions (14) and (15). In asynchronous CA–models the property is the consequence of binomial probability distribution of cells chosen for local operator application.

2.3 Operations on Cellular Arrays

When a phenomenon under simulation consists of several interacting processes, its CA–model should be composed of a number of CAs which have to interact, executing some operations on the intermediate results both on the local and global level. The problem in performing such an operation emerges when it turns to be incompatible with the alphabet of the CA–models under composition. For example, Boolean cellular arrays are incompatible with arithmetic addition. To provide such a compatibility a number of transformations on cellular arrays should be introduced, allowing to construct a kind of *CA composition algebra* [28]. on cellular arrays sets.

Like in any algebraic system, unary and binary operations are defined in CA composition algebra.

Unary Operators on Cellular Arrays

Two unary operators are defined: 1) *averaging* which transforms Boolean cellular arrays into the equivalent real ones, and 2) *state discretization*, which performs the inverse operation.

Averaging of the Boolean cellular array $\mathrm{Av}(\Omega_B)$ is a unary global operator which comprises the application of a local operator $\mathrm{Av}(m)$ to all cells of the cellular array, i.e. $\mathrm{Av}(\Omega_B) \in A_R \times M$, where $\Omega_B = \{(v, m) : v \in \{0, 1\}, m \in M\}$, $\Omega_R = \{(u, m) : u \in [0, 1], m \in M\}$.

The local operator $\mathrm{Av}(m)$ computes the average value of a cell state in the *averaging area*,

$$S_{\mathrm{Av}}(m) = \{(u_0, m), (u_1, \varphi_1(m)), \ldots, (u_q, \varphi_q(m))\}, \tag{18}$$

In case of 2D Cartesian cellular array, its underlying temolate is $T_{\mathrm{Av}}(i, j) = \{(i, j), (i + k, j + l) : k, l = -r, \ldots, r\}$, r being referred to as *averaging radius* .

Averaging may be regarded as a local operator

$$\mathrm{Av}(m) : (v,(i,j)) \star S_{\mathrm{Av}}(m) \to (u,m), \qquad u(m) = \langle v \rangle = \frac{1}{q}\sum_{k=0}^{q} v_k, \tag{19}$$

where $S_{\mathrm{Av}}(m)$ is the averaging context. Angle brackets in (19) and further denote the averaged state values.

Discretization of a real cellular array $\mathrm{Dis}(\Omega_R)$ is a unary global operator

$$\mathrm{Dis}(\Omega_R) \in A_B \times M,$$

resulting from the application of a local operator $\mathrm{Dis}(m)$ to all cells of the cellular array. $\mathrm{Dis}(m)$ is a single-cell local operator which replaces a real state value $u \in [0,1]$ by 1 with probability $p = u$.

$$\mathrm{Dis}(m) : (u,m) \to (v,m), \quad v = Bool(u) = \begin{cases} 1, & \text{if } u < rand, \\ 0 & \text{otherwise,} \end{cases} \tag{20}$$

where *rand* is a random number in the interval $[0,1]$, $Bool(u)$ means a discretized value of $u \in [0,1]$. The above two unary operations are in the following relationship.

$$\begin{array}{ll} \mathrm{Dis}(\Omega_B) = \Omega_B, & \mathrm{Dis}(\mathrm{Av}(\Omega_B)) = \Omega_B, \\ \mathrm{Av}(\Omega_R) = \Omega_R, & \mathrm{Av}(\mathrm{Dis}(\Omega_R)) = \Omega_R. \end{array} \tag{21}$$

Binary Operators on Cellular Arrays

Binary operators are defined on cellular arrays $\Omega \in A_B \times M_1 \cup A_R \times M_2$, if between $M_1 = \{(m_i)_1\}$, and $M_2 = \{(m_i)_2\}$ there exists an one-to-one correspondence $\xi : M_1 \to M_2$,

$$\begin{array}{ll} (m_i)_2 = \xi((m_i)_1), & \forall (m_i)_2 \in M_2, \\ (m_i)_1 = \xi^{-1}((m_i)_2), & \forall (m_i)_1 \in M_1. \end{array} \tag{22}$$

The cells $(v,((m_i)_1)) \in \Omega_1$ and $(u,((m_i)_2)) \in \Omega_2$ are further denoted as (v_i, m_1) and (u_i, m_2), respectively, which means that v_i and u_i are states in the corresponding cells of Ω_1 and Ω_2.

Binary operations are based on the following principle: *ordinary arithmetic rules should be valid for the averaged forms of the operands*, i.e.,

$$\Omega_1 \diamondsuit \Omega_2 \Leftrightarrow \mathrm{Av}(\Omega_1) \diamond \mathrm{Av}(\Omega_2), \tag{23}$$

where $\diamondsuit$ stands for cellular array addition $\oplus$, cellular array subtraction $\ominus$ or cellular array multiplication $\otimes$, and $\diamond$ stands for arithmetical $+$, $-$, and $\times$, respectively.

Condition (23) may also be given for the cell states as follows.

$$v_i((m_i)_1) \diamondsuit u_i((m_i)_2) \Leftrightarrow \langle v_i((m_i)_1) \rangle \diamond \langle u_i((m_i)_2) \rangle \quad \forall i \in 1, \ldots, |M|. \tag{24}$$

The reason for taking averaged state values as a generalized alphabet is twofold: 1) to allow ordinary arithmetics to be used for modeling spatial functions interactions, and 2) to make the results more comprehensive from the physical point of view.

From (23) and (24) it follows that when all operands have real alphabets, the cellular array arithmetic coincides with the corresponding real cell-by-cell *arithmetical* rules. Otherwise the rules depend on the operands alphabets.

Let $\Omega_1 = \{(v_i, m_1) : v_i \in A_1, m_1 \in M_1\}$ and $\Omega_2 = \{(u_i, m_2) : u_i \in A_2, m_2 \in M_2\}$ be the operands and $\Omega_3 = \{(w_i, m_3) : w_i \in A_3, m_3 \in M_3\}$ be a result, then binary operations are as follows.

Cellular array addition $\Omega_1 \oplus \Omega_2 = \Omega_3$. For different operands alphabets the cellular addition looks somewhat different. The following cases are of main importance.

1. Both operands Ω_1 and Ω_2 are Boolean cellular arrays, and the resulting Ω_3 is wanted to have a real alphabet. Then according to (23) Ω_3 is computed as follows.

$$\begin{array}{l} \Omega_3 = \mathrm{Av}(\Omega_1) \bigoplus \mathrm{Av}(\Omega_2), \\ w_i = \langle v_i \rangle + \langle u_i \rangle \qquad \forall i = 1 \ldots, |M|. \end{array} \tag{25}$$

2. Both operands are Boolean and the resulting cellular array is wanted to have Boolean alphabet. Then

$$\begin{array}{l} \Omega_3 = \mathrm{Dis}(\mathrm{Av}(\Omega_1) \oplus \mathrm{Av}(\Omega_2)), \\ w_i = \begin{cases} 1 & \text{if } rand < (\langle u_i \rangle + \langle v_i \rangle) \\ 0 & \text{otherwise} \end{cases} \qquad \forall i = 1, \ldots, |M|. \end{array} \tag{26}$$

3. Both operands and their sum are Boolean, the latter being used as an intermediate result. Then, it is convenient to update one of the operands, say Ω_2, so, that it be equal to the resulting array, i.e.,

$$\Omega_2(t+1) = \Omega_1(t) \oplus \Omega_2(t).$$

In that case it suffices to invert a number of zero-states in the cells $(0, m_2) \in \Omega_2$. It should be done in such a way, that in every cell $(u_i, m_2) \in \Omega_2$ its averaged state value be increased by $\langle v_i \rangle$. According to (20) the probability of such an inversion is the relation of the averaged amount of "ones" to be added to the averaged amount of "zeros" in the averaging area of each cell of Ω_2.

$$u_i' = \begin{cases} 1, & \text{if } u_i = 0 \ \& \ rand < \frac{\langle v_i \rangle}{1 - \langle u_i \rangle} \\ u_i & \text{otherwise} \end{cases} \qquad \forall i = 1, \ldots, |M|. \tag{27}$$

4. The operands have different alphabets.Let Ω_1 be a Boolean cellular array, Ω_2 – a real one, and Ω_3 is wanted to have the real alphabet. Then

$$\begin{array}{l} \Omega_3 = \mathrm{Av}(\Omega_1) \oplus \Omega_2, \\ w_i = \langle v_i \rangle + u_i, \end{array} \qquad \forall i = 1, ..., |M|. \tag{28}$$

5. Ω_1 has Boolean alphabet, Ω_2 has a real one, and Ω_3 is wanted to be a Boolean real cellular array. Two ways are possible: 1) to discretize Ω_3, obtained by (28), and 2) to update Ω_1 by using the following operation

$$w_i = \begin{cases} 1 & \text{if } v_i = 0 \ \& \ rand < \frac{u_i}{1 - \langle v_i \rangle}, \\ v_i & \text{otherwise,} \end{cases} \qquad \forall i = 1, \ldots, |M|. \tag{29}$$

Cellular array subtraction $\Omega_3 = \Omega_1 \ominus \Omega_2$. The following cases are worth to be considered.

1. Both operands are Boolean, the result is wanted to be real or Boolean. The operations are similar to those of the cellular addition. It is merely needed to replace "+" by "-" in (25) or (26).

2. Both operands are Boolean, and Ω_2 is to be updated to obtain $\Omega_2 = \Omega_1 \ominus \Omega_2$. In that case some cell states $(1, m_2) \in \Omega_2$ should be inverted with probability equal to the relation of the amount of "ones" to be removed, to the total amount of "ones" in the averaging area.

$$u_i' = \begin{cases} 0 & \text{if } u_i = 1 \ \& \ rand < \frac{\langle v_i \rangle}{\langle u_i \rangle} \\ u & \text{otherwise} \end{cases} \qquad \forall i = 1, \ldots, |M|. \tag{30}$$

3. Ω_1 has Boolean alphabet, Ω_2 has a real one, and Ω_3 is wanted to be a Boolean cellular array. Two ways are possible: 1) to discretize Ω_3, obtained by arithmetic subtraction, performing the following operation

$$\Omega_3 = \mathrm{Dis}(\mathrm{Av}(\Omega_1) - \Omega_2), \tag{31}$$

2) to update Ω_1 as follows

$$v_i' = \begin{cases} 0 & \text{if } u_i = 1 \ \& \ rand < \frac{v_i}{\langle u_i \rangle} \\ u_i & \text{otherwise,} \end{cases} \qquad \forall i = 1, \ldots, |M|. \tag{32}$$

Cellular array multiplication $\Omega_3 = \Omega_1 \otimes \Omega_2$. The operation is defined on real cellular arrays. The cell states are computed according (25) with "$\times$" instead of +. If any or both of the operands are Boolean, they should be averaged beforehand. The operation is used in those cases when one of the two operands is a constant cellular array, i.e., such one where all cell states have the same value. This is helpful when subsets of cells have to be masked or scaled.

Since addition and subtraction are defined on cellular arrays with the alphabet restricted by the interval [0,1], the same condition should be satisfied for all cells in the resulting cellular arrays. If it is not so, the alphabet is to be renormalized.

Having the set of operation on cellular arrays in hands, it is possible to formulate CA composition techniques. General composition principles prescribe to distinguish sequential, parallel, and intermixed composition techniques. Sequential composition represents several CAs processing one and the same cellular array by alternating their application at each iteration. Parallel composition suggests each CA to process its own cellular array, albeit having neighborhoods in the others. Both sequential and parallel types of composition have their versions for local and global level of operation. The global composition techniques require each CA to be applied to a result of global operator application. The local composition allows to apply local operators of different CA in any order during an iteration,

3 The Sequential Composition Techniques

Sequential composition represents a common functioning of several CA, referred to as *components*. Their local operators are applied in a certain order to one and the same cellular array. This type of composition is conceptually identical to *CA superposition*. It comprises a number of techniques differing in ordering component operators application to the cellular array under processing.

Global superposition suggests the synchronous alternation of global operators application to the component CA. When those operators use different alphabets, their compatibility should be provided by transforming a Boolean cellular array into a real one or vice versa. Apart of the general case of global superposition, two particular cases are worth to be distinguished: 1) self-superposition, which is in fact a multistage mode of a CA operation, and 2) a so-called trivial sequential composition [22] which is the superposition of the component CA evolutions.

Local superposition is the composition when at each iteration the local operators of all components involved in the composition, are applied in any order or in random. Naturally, the components should be asynchronous CA.

3.1 Global Superposition

A number of CA form a global superposition $\aleph = \Psi_{Gl}(\aleph_1, \ldots, \aleph_n)$, $\aleph = \langle A_k, M, \hat{M}_k, \theta_k, \rho_k \rangle$, if its global operator $\Phi(\Omega)$ is the result of sequential application of the global operators Φ_k to $\Omega_k = \Phi_{k-1}(\Omega_{k-1})$, $k = 1, \ldots, n$, providing compatibility of A_k and A_{k-1}, i.e.,

$$\Phi(\Omega) = \Phi'_n(\Phi'_{n-1}(\ldots \Phi'_1(\Omega_1))), \tag{33}$$

each Φ'_k being itself a superposition of Φ_k and a unary operator, i.e.,

$$\Phi'_k = \Phi_k(\mathrm{Un}(\Omega_k)), \tag{34}$$

where

$$\mathrm{Un}(\Omega_k) = \begin{cases} \mathrm{Av}(\Omega_k), & \text{if} \quad A_k = [0,1] \ \& \ A_{k-1} = \{0,1\}, \\ \mathrm{Dis}(\Omega_k), & \text{if} \quad A_k = \{0,1\} \ \& \ A_{k-1} = [0,1], \end{cases}$$

Components of the superposition may differ in alphabets, local operators and modes of operating, but the same naming set should be used.

The following particular cases of global superposition are of especial importance: self-superposition, trivial superposition, and the general type of superposition of CA with different types of alphabets.

Global Self-Superposition

This type of composition is the most simple one, being defined only for synchronous CA. A CA $\aleph = \langle A, M, \hat{M}, \theta, \sigma \rangle$ is a self-superposition $\aleph = \Psi_{SS}(\aleph_1, \ldots, \aleph_n)$, $\aleph_k = \langle A, M, \hat{M}_k, \theta, \sigma \rangle$, if its components differ only in active subsets $\hat{M}_k$.

Since the same local operator is applied at all stages of the superposition, there is no need to take care about their compatibility, so,

$$\Phi(\Omega) = \Phi_n(\Phi_{n-1}(\ldots(\Phi_1(\Omega)))).$$

Self-superposition is usually obtained by modifying a synchronous CA–model of a process which requires several neighboring cells to be updated at once. In that case non-contradictoryness condition (14) may be violated, hence, conflicts and data loss are possible. To avoid such a situations some amount of cellular parallelism should be sacrificed by performing the global transition in several stages. It is done as follows.

1. Each t-th iteration is divided into n stages $t_1(t), \ldots, t_n(t)$, the results of k-th stage being $\Omega(t_k)$.

2. At the $t_k(t)$-th stage, $\theta(m)$ is applied to all cells named $m_k \in \hat{M}_k$ of $\Omega(t_k(t))$.

A bright manifestation of a synchronous self-superposition is a well known two-stage CA–model of diffusion, which is described and studied in [1, 3, 10].

Example 1. Diffusion is a random wandering of particles aiming to even distribution. The process may be simulated by the exchange of cell states in any pair of adjacent cells. Since synchronous simulation of such a process is contradictory, as it is shown in Sect. 2.2, self–superposition of two CA: $\aleph_1 = \langle A, M, \hat{M}_1, \theta, \sigma \rangle$ and $\aleph_2 = \langle A, M, \hat{M}_2, \theta, \sigma \rangle$, is used, where $A = \{0,1\}$, $M = \{(i,j) : \ i,j = 0,1,\ldots,N\}$,

$$\begin{aligned} \hat{M}_1 &= \{(i,j) : i_{mod2} = 0, j_{mod2} = 0\}, \\ \hat{M}_2 &= \{(i,j) : i_{mod2} = 1, j_{mod2} = 1\}, \end{aligned} \tag{35}$$

$\hat{M}_1$ and $\hat{M}_2$ being referred to as *even active subset* and *odd active subset*, respectively. The local operator is as follows.

$$\theta : \{(v_0,(i,j)),(v_1,(i,j+1)),(v_2,(i,j+1)),(v_3,(i,j+1))\} \qquad (36)$$
$$\rightarrow \{(u_0,(i,j)),(u_1,(i,j+1)),(u_2,(i,j+1)),(u_3,(i,j+1))\},$$

where

$$u_k = \begin{cases} v_{(k+1)(mod4)} \text{ if } rand < p, \\ v_{(k-1)(mod4)} \text{ if } rand > (1-p), \end{cases} \qquad k = 0,1,2,3,$$

the probability p depending on the diffusion coefficient.

Each iteration of a composed CA is divided into two stages: even stage and odd stage. At the odd stage θ is applied to all cells from $\hat{M}_1$, at the even stage θ is applied to all cells from $\hat{M}_2$.

In Fig. 4 three snapshots are shown of the CA evolution simulating the diffusion of a black dye slopped onto the water surface. .

$t = 0$ $t = 4$ $t = 8$

Fig. 4. Three snapshots of diffusion process, simulated by the CA–model with a local operator (36). Black pixels stand for $\langle v \rangle = 1$, white pixels – for $\langle u \rangle = 0$.

Global Trivial Superposition

Trivial superposition $\aleph = \Psi_{Tr}(\aleph_1, \ldots, \aleph_n)$, where $\aleph_k = \langle A_k, M, \hat{M}_k, \theta_k, \rho_k \rangle$, suggests the evolution of $\aleph$ be a sequential composition of the evolutions $\Sigma_{\aleph_k}(\Omega'_k(\hat{t}_k))$ of its components, $\Omega'_k(\hat{t}_k)$ being a result of a unary operator (34) application to $\Omega_k(\hat{t}_k)$, if A_k and A_{k+1} are incompatible. The alphabets, local operators, modes of operation, and active naming subsets in the components may be different. But the order of component application is essential.

Example 2. Pattern formation process starts in the cellular array which has been obtained by a short-time application of a diffusion CA to a cellular array with two areas of high concentration (black bands along vertical borders) and empty (white) background (Fig. 5(**a**)). Diffusion is simulated by an asynchronous probabilistic CA, called in [1] a naive diffusion $\aleph_1 = \langle A, M, \hat{M}, \theta_1, \alpha \rangle$. Pattern formation is simulated by synchronous CA $\aleph_2 = \langle A, M, \hat{M}, \theta_2, \sigma \rangle$. Both CA have a Boolean alphabet $A = \{0,1\}$, their naming sets are identical as well as active naming subsets, $M = \hat{M} = \{(i,j) : i,j = 0,\ldots,300\}$.

The local operator of the CA $\aleph_1$ is as follows.

$$\begin{aligned}\theta_1 : &\{(v_0,(i,j)),(v_1,(i-1,j)),(v_2,(i,j+1)),(v_3,(i+1,j)),(v_4,(i-1,j))\}\\ \rightarrow\ &\{(u_0,(i,j)),(u_1,(i-1,j)),(u_2,(i,j+1)),(u_3,(i+1,j)),(u_4,(i-1,j))\}\end{aligned} \tag{37}$$

with the transition functions

$$\begin{aligned} u_0 &= v_k, \quad \text{if} \quad 0.25k < rand < 0.25(k+1),\\ u_k &= \begin{cases} v_0 & \text{if} \quad 0.25k < rand < 0.25(k+1), \\ v_k & \text{otherwise.} \end{cases} \qquad k = 1,\ldots,4. \end{aligned} \tag{38}$$

The local operator in the pattern formation CA $\aleph_2$ is as follows.

$$\theta_2 : (u_0,(i,j)) \star \{(u_l,\phi_l(i,j)) : l = 1,\ldots,q\} \rightarrow (v_0,(i,j)), \tag{39}$$

where $q = (2r+1)^2$, $r = 3$ being the radius of neighborhood template, $q = 49$.

$$\phi_l(i,j) = (i+g_l, j+h_l), \quad g_l = l_{mod(2r+1)} - r, \quad h_l = \lfloor l/(2r+1) \rfloor;$$

$$v_0 = \begin{cases} 1, & \text{if} \quad \sum_{l=0}^{q} w_l v_l > 0 \\ 0, & \text{otherwise,} \end{cases} \tag{40}$$

where

$$w_l = \begin{cases} 1, & \text{if} \quad g_l \le 1 \ \& \ \ h_l \le 1 \\ -0,2 & \text{otherwise.} \end{cases}$$

Sum in (40) may be also obtained by imposing weighted template

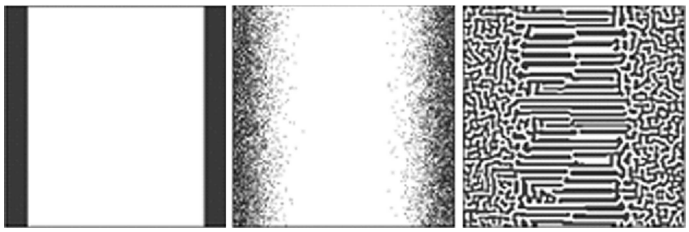
$$\mathrm{W} = \begin{array}{|c|c|c|c|c|c|c|} \hline a & a & a & a & a & a & a \\ \hline a & a & a & a & a & a & a \\ \hline a & a & 1 & 1 & 1 & a & a \\ \hline a & a & 1 & \mathbf{1} & 1 & a & a \\ \hline a & a & 1 & 1 & 1 & a & a \\ \hline a & a & a & a & a & a & a \\ \hline a & a & a & a & a & a & a \\ \hline \end{array} \quad , \qquad a = \text{-0.2},$$

onto a cell and computing the sum of products of its entries and underlying cell states.

In Fig.5 three snapshots of trivial composition of two CA ($\aleph_1$ simulating diffusion and $\aleph_2$ simulating pattern formation) are shown. Cellular array size is 300×300, $\hat{t}_1 = 10$, $\hat{t}_2 = 12$. The obtained pattern is a stable one, further application of θ_2 to $\Omega_2(\hat{t}_2)$ implies no change in it.

Global Superposition of Arbitrary CA

Global superposition of arbitrary CA is a technique for obtaining a CA $\aleph_{Gl} = \Psi_{Gl}(\aleph_1,\ldots,\aleph_n)$, which combines operation of several CA $\aleph_k = \langle A_k, M, \hat{M}_k\theta_k, \rho\rangle$, $k = 1,..,n$, whose alphabets and local operators are allowed



a) **b**) **c**)

Fig. 5. Three snapshots of the process, simulated by trivial superposition of an asynchronous diffusion CA and a synchronous pattern formation CA: **a**) initial array, **b**) $\hat{t}_1 = 10$, **c**) $\hat{t}_2 = 12$.

to be incompatible, and modes of operation may be different. The operation of the composed CA is as follows.

1. Each t-th iteration of the composed CA consists of n stages $t_1(t), \ldots, t_n(t)$, the results of t_k-th stage being $\Omega(t_k(t)) = \Phi_{k-1}(t_{k-1}(t))$.

2. At the $t_k(t)$-th stage θ_k is applied to all cells $m \in \hat{M}_k$ of $\Omega'(t_k(t))$. The latter should be obtained by transforming $\Omega(t_k(t))$ according to (34), if needed.

Example 3. Simulation of the alga spreading over the water is considered to combine three elementary processes:

1) agglomeration of randomly distributed alga,

2) diffusion of alga into water,

3) procreation of alga .

The first process is represented by a Boolean CA $\aleph_1$ sometimes called a phase-separation CA [31, 22], the second — by the two-stage diffusion CA $\aleph_2$ given in Example 1 (Sect. 3.1), the third — by $\aleph_3$ computing a nonlinear logistic function [32] in each cell. Accordingly, each tth iteration of the composed CA has three stages, i.e.,

$$\Omega(t) \to \Omega(t_1) \to \Omega(t_2) \to \Omega(t_3), \quad \Omega(t_3) = \Omega(t+1)$$

.

At the first stage t_1, the transition $\Omega(t) \to \Omega(t_1)$ is performed by a synchronous CA $\aleph_1 = \langle A_1, M, \hat{M}_1, \theta_1, \sigma \rangle$ with $A_1 = \{0, 1\}$, $M = \{(i, j) : i, j = 0, \ldots, N\}$, $\hat{M} = M$, and a single-cell updating local operator

$$\theta_1(i,j)) : (v,(i,j)) \star S''(i,j) \to \{(v',(i,j))\} \qquad \forall (i,j) \in M, \qquad (41)$$

where

$$S''(i,j) = \{(v_k, \phi_k(i,j)) : \ \phi_k(i,j) = (i+g, j+h)\}, \\ g, h \in \{-2, -1, 1, 2\},$$

and

$$v' = \begin{cases} 1, & \text{if} \quad s < 24 \text{ or } s = 25, \\ 0, & \text{if} \quad s > 25 \text{ or } s = 24. \end{cases} \quad \text{where} \quad s = \sum_{g=-2}^{2} \sum_{h=-2}^{2} v_{i+g,j+h}.$$

At the second stage $\aleph_2$ given in Example 1 performs a transition $\Omega(t_1) \to \Omega(t_2)$ by application θ_2 (36) to all cells of $\Omega(t_1)$), the value of the probability in (36) being $p = 0.5$. As the alphabet A_2 is compatible with A_1, θ_2 is applied directly to the cells of Ω_1 resulting in a Boolean array $\Omega(t_2) = \{(u, (i, j))\}$.

At the third stage alga procreation CA $\aleph_3 = \langle A_3, M, \hat{M}_k, \theta_3, \sigma \rangle$ is applied to $\Omega(t_2)$. But since $A_3 = [0, 1]$ and, hence, θ_3 is incompatible with Ω_2, the latter is transformed into $\Omega(t_2)'$ by averaging, i.e. the operator $\mathrm{Av}(i, j)$ is applied to $\Omega(t_2)$ replacing each cell state $u(i, j)$ by $\langle u(i, j) \rangle$. The latter is computed according to (19) with the averaging template $T_{\mathrm{Av}}(i, j) = \{(i + k, j + l) : k, l = -8, \ldots, 8\}$. The local operator

$$\theta_3 : (\langle u(i, j) \rangle, (i, j)) \to (F(\langle u(i, j) \rangle), (i, j)) \tag{42}$$

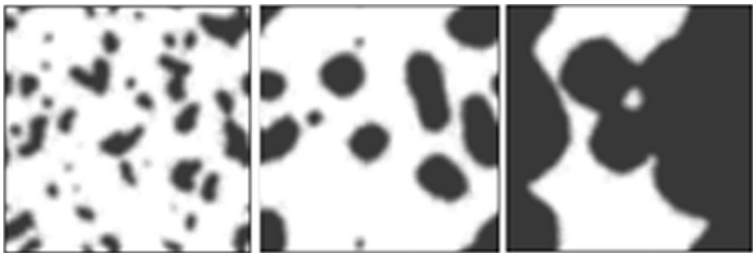
is applied to $\Omega(t_2)'$ replacing a cell state $\langle u(i, j) \rangle$ by the value of a nonlinear logistic function

$$F(\langle u(i, j) \rangle) = 0.5 \langle u(i, j) \rangle (1 - \langle u(i, j) \rangle).$$

The resulting cellular array having real states should be discretized according to (20) to obtain $\Omega(t_3)' = \Omega(t + 1)$.

The composition has been applied to an initial Boolean cellular array Ω with $v = 1$ randomly distributed with probability $p = 0.5$ so that $\langle v(i, j) \rangle \approx 0.5$ for all $(i, j) \in M$, the border conditions being periodic.

In Fig.6, three snapshots of the simulation process are shown, cellular arrays being averaged for making the observation more comprehensive. Black pixels stand for maximum concentration of alga, white ones represent clear water. It is seen that on the first iterations, the total amount of alga decreases, but if some compact spots remain large enough, the procreation activeness enhances their growth up to the saturation.



$t = 5$ $t = 25$ $t = 70$

Fig. 6. Three snapshots of alga spreading in water, simulated by synchronous global superposition of $\aleph_1$ with θ_1 (41), $\aleph_2$ with θ_2 (36) and $\aleph_3$ with θ_3 (42). Black pixels stand for maximal concentration of alga, white pixels – for clear water.

3.2 Local Superposition

Asynchronous local superposition is mainly used in simulating biological processes and nano-kinetics, i.e., the processes on micro- or nano-level, which are considered to be completely stochastic by nature. This technique aims at obtaining a CA–model $\aleph = \Psi_{Loc}(\aleph_1, \ldots, \aleph_n\}$ composed of n asynchronous CA $\aleph_k = \langle A, M, \hat{M}_k, \theta_k, \alpha\rangle$, $k = 1, \ldots, n$, which differ only in local operators and (perhaps) in active subsets. The way of their common functioning is as follows. An iteration $\Omega(t) \rightarrow \Omega(t+1)$ consists of $|M|$ cycles, a cycle being a sequence of single-shot applications of $\theta_k(m)$, $k = 1, \ldots, n$, to a randomly chosen cell from $\Omega(t)$. Each θ_k is executed immediately after the application. There is no constraints neither on the order of choosing a cell during an iteration, nor on the order of choosing θ_k for application during a cycle. Sometimes, the natural features of the process under simulation dictate a certain ordering or grouping of local operators in a cycle, but usually they are chosen in random.

Example 4. A chemical reaction of CO oxidation over platinum catalysts, well known in surface chemistry as Ziff-Guilari-Barshod model [30], is represented by a local superposition of four simple local operators, mimicking elementary actions of adsorption, reaction, oxidation, and diffusion. The cellular array Ω corresponds to a catalysts plate, each site on it being named by $(i, j) \in M$, $|M| = N \times N$, $\hat{M} = M$. The alphabet contains three symbols $A = \{a, b, 0\}$, so that $(a, (i, j))$, $(b, (i, j))$, and $(0, (i, j))$ are cells corresponding to the sites occupied by the molecules of CO, O, or being empty, respectively. In the initial array, all cells are empty. The CO oxidation process consists of the following four elementary molecular actions in any cell named (i, j).

1) Adsorption of CO from the gas: if the cell (i, j) is empty, it becomes occupied by a CO molecule with probability p_1, whose value depends on the partial pressure of CO in the gas above the plate.

2) Adsorption of the oxygen O_2 from the gas: if the cell (i, j) is empty and has an empty adjacent cell, both become occupied by an atom of oxygen with probability p_2. The probability depends on the partial pressure of oxygen in the gas. One out of $h < 4$ adjacent cells of the cell (i, j) is chosen with probability $p_n = 1/h$.

3) Reaction of oxidation of CO (CO+O $\rightarrow$ CO_2): if the cell (i, j) occurs to be in a CO state and its adjacent cell is in O state, then the molecule CO_2, formed by the reaction, transits to the gas and both cells become empty. One out of $h < 4$ adjacent cells occupied by oxygen is chosen with probability $p_n = 1/h$.

4) Diffusion of CO over the plate: if the cell (i, j) occurs to be in a CO state when one of its adjacent cells is empty, the cell (i, j) becomes empty, and the empty cell gets the state CO. This occurs with probability p_3. One out of $h < 4$ adjacent cells of the cell (i, j) is chosen with probability $p_n = 1/h$.

Chemical formulas and graphical representation of the components actions are shown in Fig.7.

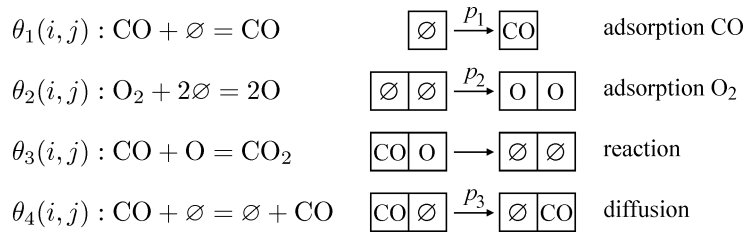


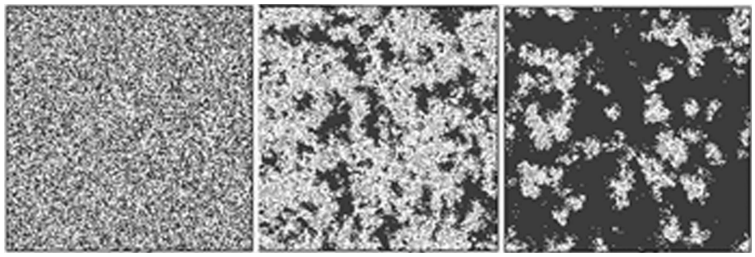
Fig. 7. Graphical representation of local operators involved in an asynchronous local superposition simulating chemical oxidation of CO on platinum

Formally, local operators of the above actions are represented as follows.

$$\begin{aligned}
\theta_1(i,j):&\ \{(0,(i,j))\} \to \{(a,(i,j))\}, \quad \text{if} \quad p_1 > rand,\\
\theta_2(i,j):&\ \{(0,(i,j))(0,\phi_k(i,j))\} \to \{(b,(i,j)),(b,\phi_k(i,j))\},\\
&\ \text{if} \quad (k-1)p_n < rand < kp_n \ \&\ p_2 > rand\\
\theta_3(i,j):&\ \{(a,(i,j))(b,\phi_k(i,j))\} \to \{(0,(i,j)),(0,\phi_k(i,j))\},\\
&\ \text{if} \quad (k-1)p_n < rand < kp_n\\
\theta_4(i,j):&\ \{(a,(i,j))(0,\phi_k(i,j))\} \to \{(0,(i,j)),(a,\phi_k(i,j))\},\\
&\ \text{if} \quad (k-1)p_n < rand < kp_n) \ \&\ p_3 > rand,
\end{aligned} \tag{43}$$

for $k = 1, \ldots, 4$.

In Fig.8 three snapshots of the simulation process are shown, the initial cellular array $\Omega(0) = \{(0,(i,j)) : \forall (i,j) \in M\}$, $|M| = 200 \times 200$.



$t = 2$ $t = 40$ $t = 80$

Fig. 8. Three snapshots of the oxidation reaction simulation by an asynchronous superposition of local operators (43). Black pixels stand for CO, gray pixels – for O, and white pixels – for empty sites

In the general case local superposition is not a commutative operation, i.e., if $\theta_1 \neq \theta_2$, then

$$\theta_1(\theta_2(m)) \neq \theta_2(\theta_1(m)). \tag{44}$$

The above property is very important, because the results of the simulation may differ essentially if the order of superpositions is changed. Although in

case of long evolution, the repetitive sequence of superpositions, for example, such as $\theta_1(\theta_2(\theta_1(\theta_2(m)\ldots)))$, makes the composition insensitive of the substitution being the first. If it is not the case, the only way to make the result independent of the order of substitutions in the composition is their random choice at any step of application (the Monte-Carlo method).

4 The Parallel Composition Techniques

Parallel composition suggests functioning of n interacting CA, each processing its own cellular array. Taking into account that the number of possible interactions in the composition exponentially increases with n, and for clearness of presentation, the composition $\aleph = \Upsilon(\aleph_1, \aleph_2)$ of not more than two CA is further considered. The components $\aleph_k = \langle A_k, M_k, \hat{M}_k, \theta_k, \rho_k \rangle, k = 1, 2$, are allowed to have different alphabets, different modes of operation, different local operators, and between $M_1 = \{(m_i)_1\}$, and $M_2 = \{(m_i)_2\}$, $i = 1, 2 \ldots, |M|$, the condition (22) is satisfied.

Since θ_1 and θ_2 are to be executed simultaneously, the computation is dangerous from the point of view of non-contradictoryness condition (14), which states that only a single updating may be performed simultaneously on one and the same cell. On the other hand, in order that the component transition function in θ_1 and θ_2 interact, they should use common state variables. Hence, with respect to (11) and (13), the left-hand sides of θ_1 and θ_2 should have nonempty intersection, i.e.

$$(S_1((m_i)_1) \cup S_1''((m_i)_1) \cap (S_2((m_i)_2) \cup S_2''((m_i)_2) \neq \emptyset.$$

Combining this statement with (14) the correctness condition for parallel composition yields

$$T_k((m_i)_k) \subseteq M_k, \tag{45}$$

$$T_k''((m_i)_k) \subseteq (M_1 \cup M_2) \quad \forall k \in \{1, 2\}. \tag{46}$$

From (45) it follows that $\theta_1((m_i)_1)$ and $\theta_2((m_i)_2)$ may update cells only from their own cellular arrays, whereas from (46) they are allowed to use cell states of the both. It means, that the neighborhoods of cells $(m_i)_1$ and $(m_i)_2$, may intersect only by their contexts.

The above conditions are valid both for local and global composition techniques, as well as both for CA with synchronous and asynchronous modes of operation.

4.1 Global Parallel Composition

Similarly to sequential case a trivial parallel CA composition also exists.

Trivial Parallel Composition

Trivial parallel composition $\aleph = \Upsilon_{Tr}(\aleph_1, \aleph_2)$, $\aleph_k = \langle A_k, M_k, \hat{M}_k, \theta_k, \rho_k \rangle$, $k = 1, 2$, is a degenerate particular case of parallel composition, when the components are completely independent, i.e. their neighborhood templates do not intersect at all, i.e.,

$$(S_1((m_i)_1) \cup S_1''((m_i)_1) \cap (S_2((m_i)_2) \cup S_2''((m_i)_2) = \emptyset. \tag{47}$$

Nonetheless, after both components have terminated to evolute, a binary operation on the resulting cellular arrays may be performed. Hence, the result of composed CA computation is

$$\Omega(\hat{t}) = \Omega_1(\hat{t}_1) \diamondsuit \Omega_2(\hat{t}_2), \tag{48}$$

where $\diamondsuit$ is any binary operator given in Sec: 2.3.

Example 5. Two phase separation models are to be compared by computing the difference of two resulting cellular arrays:

1) $\Omega_1(\hat{t}_1)$ obtained by the evolution of a totalistic CA $\aleph_1 = \langle A_1, M_1, \hat{M}_1, \sigma \rangle$, which is described in Example 3 (Sec: 3.1) with θ_1, given as (41), and

2) $\Omega_2(\hat{t}_2)$ obtained by solving a PDE proposed in [33] which describes the same process,

$$\frac{\partial u}{\partial t'} = 0.2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) - 0.2(u-1)(u-0.5)(u-0.9). \tag{49}$$

Let us consider the finite-difference representation of (49) as a synchronous CA $\aleph_2 = \langle A_2, M_2, \hat{M}_2, \theta_2, \sigma \rangle$, where $A_2 = [0, 1]$, $M_2 = \hat{M}_2 = \{(i,j)_2 : i = x/h, j = y/h, i, j = 0, \ldots, N\}$, h being a space step, $t = t'/(\Delta t)$ — iteration time,

$$\begin{aligned} \theta_2 : (u_0, (i,j)_2) \star \{(u_1, (i-1,j)_2), (u_2, (i,j+1)_2), (u_3, (i+1,j)_2), (u_4, i, j-1)_2)\} \\ \rightarrow (u_0', (i,j)_2), \end{aligned} \tag{50}$$

where

$$u_0' = \frac{u_1 + u_2 + u_3 + u_4 - 4u_0}{h^2}.$$

The initial cellular arrays $\Omega_1(0)$ and $\Omega_2(0)$ for $\aleph_1$ and $\aleph_2$ are identical, having equal distribution of "ones" and "zeros", so, that $\langle v(i,j)_1 \rangle = u(i,j)_2 = 0.5$ for all $(i,j)_1 \in M_1$ and all $(i,j)_2 \in M_2$.

The comparison of both components evolutions

$$\Omega_1(\hat{t}_1) = \{(v, (i,j)_1) : \ v \in A_1, \ (i,j)_1 \in M_1\},$$

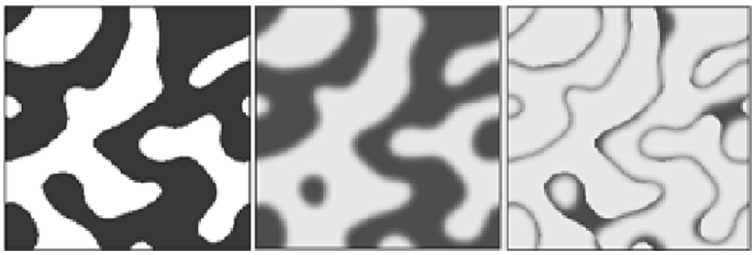
and

$$\Omega_2(\hat{t}_2) = \{(u, (i,j)_2) : \ u \in A_2, \ (i,j)_2 \in M_2\},$$

is done by computing the absolute value of their cellular arrays subtraction defined in Sec. 2.3. Since $\Omega_1(\hat{t}_1)$ and $\Omega_2(\hat{t}_2)$ are incompatible the first is to be averaged according to (19). The final result is obtained as $\Omega'_2(\hat{t}) = \{(u'_2, (i,j)_2)\}$, where

$$u'_2((i,j)_2) = |\langle v_1((i,j)_1)\rangle - u((i,j)_2)|. \tag{51}$$

The three resulting cellular arrays: $\Omega_1(\hat{t}_1)$, $\Omega_2(\hat{t}_2)$, and $\Omega'_2(\hat{t})$ are shown in Fig.9.



a **b** **c**

Fig. 9. Three snapshots of parallel trivial composition of two CA simulating phase separation: (**a**) resulting cellular array obtained by a totalistic CA (41), (**b**) resulting cellular array obtained by a CA based on PDE (50), and (**c**) their difference. Black pixels stand for 1, white for 0, gray scale intensity corresponds to values from $[0, 1]$

Nontrivial Parallel Composition

Nontrivial parallel composition $\aleph = \Psi(\aleph_1, \aleph_2)$ suggests that all components $\aleph_1$ and $\aleph_2$ interact at each iteration. Two types of interaction between them determine two types of parallel composition techniques: unidirectional parallel composition and bidirectional parallel composition [22].

In *unidirectional parallel composition*, one of the components, say $\aleph_1$, evolutes independently, i.e., the transition functions of θ_1 do not depend on the states of cells from Ω_2. But, the transition functions of $\aleph_2$ depend on states of both cellular arrays. Hence, condition (46) takes the following form.

$$T''_1((m_i)_1) \subseteq M_1, \quad \forall (m_i)_1 \in M_1, \tag{52}$$

$$T''_2((m_i)_2) \subseteq (M_1 \cup M_2) \quad \forall (m_i)_2 \in M_2. \tag{53}$$

Such a kind of composition is frequently used when simulating a certain process by $\aleph_1$, and using auxiliary CA $\aleph_2$ for transforming simulation results of $\aleph_1$ into a proper form for analyzing or visualizing its evolution. For example, $\aleph_1$ is a Boolean CA, and observation of its evolution requires it to be real numbers. Then, $\aleph_1$ evolutes independently, and $\aleph_2$ performs the averaging of $\Omega_1(t)$ at each iteration using cell states of Ω_1 in its transition functions.

In *bidirectional parallel composition* transition functions of both components depend on states of cells from both cellular arrays, i.e.

$$\begin{aligned} T_1''((m_i)_1) \subseteq (M_1 \cup M_2) \quad \forall (m_i)_1 \in M_1, \\ T_2''((m_i)_2) \subseteq (M_1 \cup M_2) \quad \forall (m_i)_2 \in M_2, \end{aligned} \tag{54}$$

(45) being preserved as well. If the alphabets of $\aleph_1$ and $\aleph_2$ are incompatible, then a suitable unary transformations of $\Omega_1(t)$ or $\Omega_2(t)$ should be done after each iteration.

Example 6. A 2D reaction–diffusion process of autocatalytic reaction propagation in a domain with obstacles is simulated by bidirectional parallel composition of two CA:

1) a two-stage synchronous diffusion CA $\aleph_1 = \langle A_1, M_1, \hat{M}_1, \theta_1, \sigma \rangle$ given in Example 1 (Sect. 3.1), and

2) a single cell synchronous CA $\aleph_2 = \langle A_2, M_2, \hat{M}_2, \theta_2, \sigma \rangle$ which computes a real nonlinear function of the cell state.

Since A_1 and A_2 are incompatible, unary operators $\mathrm{Dis}(i,j)$ and $\mathrm{Av}(i,j)$ should be added, which is done by means of incorporating them into the local operators θ_1 and θ_2, respectively. In θ_1 the operator $\mathrm{Dis}(u(i,j)_1)$ is included in the transition function as follows.

$$\begin{aligned} \theta_1 : \{(v_0,(i,j)_1),(v_1,(i,j+1)_1),(v_2,(i+1,j)_1),(v_3,(i,j-1)_1)\} \\ \star\{(u_0,(i,j)_2),(u_1,(i,j+1)_2),(u_2,(i+1,j)_2),(u_3,(i,j-1)_2)\} \\ \rightarrow \{(v_0',(i,j)_1),(v_1',(i,j+1)_1),(v_2',(i+1,j)_1),(v_3',(i,j-1)_1)\}, \end{aligned} \tag{55}$$

where

$$v_k' = \begin{cases} \mathrm{Bool}(u_{(k+1)(mod4)} & \text{if } rand < p, \\ \mathrm{Bool}(u_{(k-1)(mod4)} & \text{if } rand > (1-p), \end{cases}$$

The local operator $\theta_2((i,j)_2)$ is combined with $\mathrm{Av}((i,j)_1)$ which results in the following.

$$\theta_2 : (u,(i,j)_2) \star \{S_{\mathrm{Av}}((i,j)_1)\} \rightarrow f(\langle v((i,j)_1)\rangle,(i,j)_2), \tag{56}$$

where

$$f(\langle v((i,j)_1)\rangle) = 0.5\langle v((i,j)_1)\rangle(1-\langle v((i,j)_1)\rangle),$$

$\langle v(i,j)_1)\rangle$ being obtained according to (19).

The simulation space is a square area 300×300 cells with a number of rectangular obstacles through which the reaction cannot penetrate (in Fig.10 shown in dark gray). The initial cellular array $\Omega_1(0)$ has a little spot of a reactant of high density in the central part of the top side of the array. The reaction product propagation is shown in Fig.10.

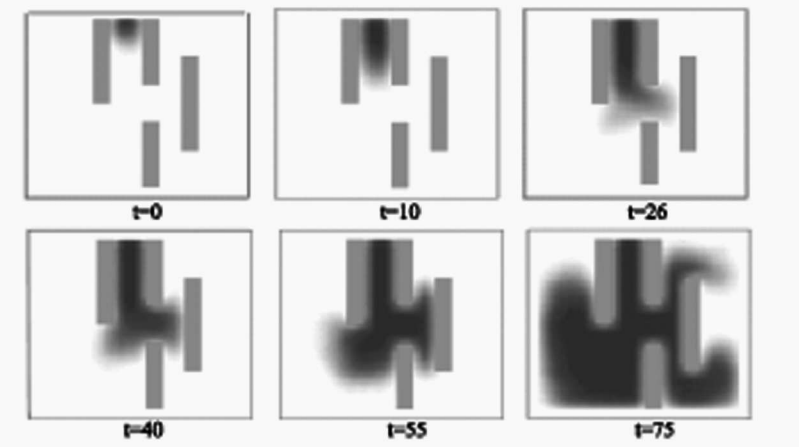


Fig. 10. Three snapshots of $\aleph_2$ evolution of a parallel bidirectional composition simulating the front propagation of autocatalytic reaction. Black pixels stand for obstacles, grey pixels – for maximal concentration of the reactant, white – for reactant absence

4.2 Local Parallel Composition

Like in sequential case this type of composition aims at obtaining an asynchronous CA–model $\aleph = \Upsilon_{Loc}(\aleph_1, \aleph_2)$ composed of two asynchronous CA $\aleph_k = \{A_k, M_k, \hat{M}_k, \theta_k, \alpha\}$, $k = 1, 2$. The components may differ in alphabets and in local operators, naming sets M_1 and M_2 being in the relation (22). The way of the composed CA functioning is as follows.

Both components operate in parallel in asynchronous mode: at each tth iteration the local operator θ_k is applied to all cells of $\hat{M}_k$, the cells being selected in any order and updated immediately after selection. All variations of cell selection ordering are allowed, namely the following ones are possible.

1) Random selection of cells in both CA independently according to given probability distribution, usually the binomial distribution is used.

2) Random selection of cells but one and the same for both CA, the cells $(m_i)_1$ and $(m_i)_2 = \xi(m_i)_1$ being updated simultaneously.

3) Determinate prescribed order of cell selection, one and the same in both CA.

4) Alternation of θ_1 and θ_2 application with any order of cells selection.

Like in the global parallel composition case, the local parallel composition may be unidirectional and bidirectional, depending on the number (one or both) of components having local configurations contexts in the cellular array of the other component.

Example 7. A soliton-like 1D process is simulated by a parity totalistic CA [4] $\aleph_1 = \{A_1, M_1, \hat{M}_1, \theta_1, \alpha\}$. Since A_1 is a Boolean alphabet the process

is difficult to recognize as two moving waves passing one through the other. So, to make the process observable in a habitual form, $\aleph_1$ is combined with another CA $\aleph_2 = \{A_2, M_2, \hat{M}_2, \theta_2, \alpha\}$ which performs averaging of any cell state in Ω_1 just after its updating. Thus, the composition of $\aleph_1$ and $\aleph_2$ is local unidirectional: $\aleph_1$ operates independently, and $\aleph_2$ uses cell states of Ω_1 as the context in θ_2. The naming sets $M_1 = \hat{M}_1 = \{i_1 : i = 0, \ldots, N\}$, and $M_2 = \hat{M}_2 = \{i_2 : i = 0, \ldots, N\}$, are in one-to one correspondence, i.e. $i_2 = \xi((i_1))$. The local operator

$$\begin{aligned} \theta_1 : (v_0, i_1) \star \{(v_j, i_1 + j) : \ j = -r, \ldots, -1, 1, \ldots, r\} \\ \rightarrow (v'_0, i_1), \end{aligned} \tag{57}$$

where

$$v'_0(i_1) = \begin{cases} 1, & \text{if} \quad w \neq 0 \ \& \ w = 0_{mod2} \\ 0, & \text{otherwise}, \end{cases} \tag{58}$$

$$w = \sum_{j=-r}^{r} v_j.$$

The mode of $\aleph_1$ operation is an ordered asynchronous one: θ_1 is applied sequentially according to the cell numbers $i_1 = 0, 1, \ldots N$, each cell (v, i_1) being immediately updated, so, that the cell states $v(i_1 + j)$ with $j < 0$ situated leftwards of i_1, are already in the next state, while the rightward cells are yet in the current state. Border conditions are periodic. The initial global cellular state $\Omega_1(0)$ has certain patterns referred to as "particles" [4]. Here, the two following particles are used: $P_1 = 1101$, and $P_2 = 10001001$ with $r = 4$. All others cells are in zero states. The evolution of $\aleph_1$ shows that the first particle P_1 appears in $\Omega_1(t)$ any 2 iterations being displaced by $d_1 = 7$ cells to the left. And the second particle P_2 appears in $\Omega_1(t)$, any 6 iteration being displaced by $d_2 = 12$ cells also to the left. So, each 6 iterations P_1 is displaced by 21 cells, and P_2 — by 12 ones. Hence, the distance between the particles diminishes by 9 cells in each 6 iterations. After the start ($t = 0$) during the period from $t = 12$ till $t = 24$ the particles are superimposed, and after $t = 30$ the first particle is ahead, as it is shown in the following global states.

$t = 0$: 0000 . . . 000000000000**10001001**0000000000000000000000000**11011**00
$t = 6$: 0000 . . . 00**10001001**00000000000000**1101**1000000000000000000000
$t = 30$: 000000000000000000000000**1101**1000**10001001**00 . . . 0000000000000
$t = 36$: 00**1101**1000000000000000**10001001**00000 . . . 0000000000000000000

The second CA $\aleph_2$ performs an asynchronous averaging of Ω_1, in order to transform patterns displacement into waves propagation. The steps of $\aleph_2$ are synchronized with those of $\aleph_1$ and the order of cell selection is the same.

$$\begin{aligned} \theta_2 : (v_0, i_2) \star \{(v_j, i_1 + j) : \ j = -r, \ldots, -1, 1, \ldots, r\} \\ \rightarrow (\langle v_0 \rangle, i_2), \end{aligned} \tag{59}$$

where

$$\langle v_0 \rangle = \frac{1}{(2r+1)} \sum_{j=-r}^{r} v_j.$$

In Fig.11 three snapshots of Ω_2 are shown.

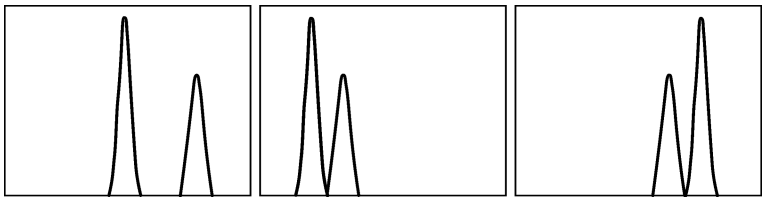


Fig. 11. Three snapshots of the soliton propagation obtained by simulating the process using local parallel composition of two CA with local operators given by (57) and (59)

4.3 Mixed Composition

In practice, complex phenomena simulation requires a number of CA–models to be included in a composition forming a complicated scheme of different composition techniques. The main principle for constructing such a mixed composition is that any component may be itself a composed CA. Hence, mixed composition is a hierarchical structure, any level of hierarchy being a composed CA,

Example 8. A simplified process of vapor nucleation in binary system (vapor, gas–carrier) is simulated using a mixed CA composition. The process has been studied in a number of investigations on self-organizing reaction-diffusion systems. For example, in [34] an attempt is made to solve the PDE system which describes the process as follows.

$$\begin{aligned} \frac{\partial v}{\partial t} &= 0.025\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + 0.2v - v^3 - 1.5u, \\ \frac{\partial u}{\partial t} &= 0.0025\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + v - u. \end{aligned} \tag{60}$$

Since two species are involved in the process, a bidirectional parallel composition should be used. The resulting CA $\aleph = \Upsilon(\aleph_1, \aleph_2)$ has two components, each simulating a reaction-diffusion process in $\Omega_1 = \{(v, (ij)_1)\}$ (vapor) and $\Omega_2 = \{(u, (ij)_2)\}$ (gas), respectively. Each component $\aleph_k = \Psi_{Gl}(\aleph_{Dk}, \aleph_{Rk})$,

in its turn, is a sequential composition of $\aleph_{Dk} = \langle A_D, M_k, \hat{M}_k, \theta_{Dk}, \beta\rangle$ which represents the diffusion, and $\aleph_{Rk} = \langle A_R, M_k, \hat{M}_k, \theta_{Rk}, \sigma\rangle$, which represents the reaction. The two diffusion CA, $\aleph_{D1}$ and $\aleph_{D2}$, operate each in its own cellular array independently. Their results are used by the reaction CA $\aleph_{R1}$ or $\aleph_{R2}$, which are in the bidirectional parallel composition with each other. Since the alphabets A_D and A_R are incompatible, the diffusion global operator result $\Phi_{Dk}(\Omega_{D_k}(t))$ is averaged, and that of the reaction $\Phi_{Rk}(\Omega_{R_k}(t))$ is discretized, which yields the following superposition of global operations of $\aleph_k$.

$$\Phi_k(\Omega_k(t)) = \mathrm{Dis}(\Phi_{Rk}(\mathrm{Av}(\Phi_{Dk}(\Omega_k(t-1))))), \quad k = 1, 2. \tag{61}$$

Diffusion is simulated by the two-stage synchronous CA given in Example 1 (Sect.3.1) with θ_{Dk} given as (36). The difference between $\aleph_{D1}$ and $\aleph_{D2}$ is in the values of probabilities used in the transition function. They are: $p_v = 0.5$, $p_u = 0.05$, which corresponds to the diffusion coefficients in (60), provided the time step $\Delta t = 0,6$ sec and space step $h = 0.1$ cm.

Reaction is simulated by a single cell context-free CA with the following local operators.

$$\begin{aligned} &\theta_{R1} : (\langle v\rangle, (i,j)_1) \to (f_v(\langle v((i,j)_1)\rangle, \langle u((i,j)_2)\rangle), (i,j)_1), \\ &\theta_{R2} : (\langle u\rangle, (i,j)_2) \to (f_u(\langle v((i,j)_1)\rangle, \langle u((i,j)_2)\rangle), (i,j)_2), \end{aligned} \tag{62}$$

where

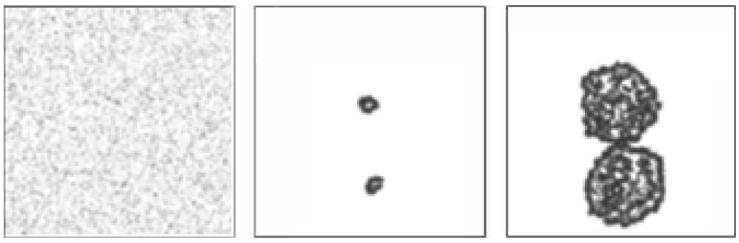
$$\begin{aligned} f_v &= 0.2\langle v((i,j)_1)\rangle - \langle v((i,j)_1)\rangle^3 - 1.5\langle u(i,j)_2)\rangle, \\ f_u &= \langle v((i,j)_1)\rangle - \langle u((i,j)_2)\rangle, \end{aligned}$$

The size of both cellular arrays is 300×300 cells with periodic border conditions. The initial conditions are Boolean cellular arrays with the following evenly distributed concentrations of vapor and gas: $\langle v(i,j)_1\rangle = 0.1$, $\langle v(i,j)_2\rangle = 0.9$.

The evolutions of $\aleph_1$ and $\aleph_2$ show the processes of vapor and gas space-time distribution, respectively. In Fig. 12 three snapshots are shown for vapor nucleation process. Emergency of small vapor bubbles from a fog is observed. The bubbles grow in size exhibiting oscillations of vapor density inside them.

5 Computational Properties of Composed CA

In real simulation tasks when dealing with large CA size and large amount of iterations, the computational properties, such as accuracy, stability, and complexity are of main importance. Hence, the impact of above composition techniques on these properties should be assessed. As for the accuracy, the study of this property is focused on the procedures which are beyond the conventional cellular automata theory, namely, cellular array transformations for



$\Omega_1(0)$ $\Omega_1(12)$ $\Omega_1(40)$

Fig. 12. Three snapshots of vapor nucleation process obtained by simulating it as a parallel composition of two superpositions or diffusion and reaction. Black pixels stand for vapor particles.

providing compatibility, since it is precisely these operations that may contribute some errors. Stability assessment of the composition directly depends on the stability of its components, which may exhibit different kind of behavior [2], their evolutions tending to a stable state or never reaching it, or being chaotic. So, the attention is focused on stability conservation, provided the components of CA composition are stable. The property of complexity is concerned with the additional operations which are incorporated to eliminate incompatibility between the interacting components.

It should be noticed that contemporary mathematics has no well–established concepts of CA computational properties, as well as no methods for their quantitative assessment. So, the subsections below may be regarded as some considerations for the problem, indicating the points for further investigation.

5.1 Accuracy of the Composed CA

One of Boolean CA advantages is that they are absolutely accurate from the computational standpoint, i.e. no errors are incorporated by rounding off. But, once averaging $\mathrm{Av}(\Omega)$ or discretization $\mathrm{Dis}(\Omega)$ is used and, hence, real numbers are processed, the errors may be brought in.

In trivial compositions, both sequential and parallel ones, the two above operations are performed only once at the start and at the end of the simulation process, bringing in inessential approximation error. But in nontrivial compositions, when $\mathrm{Av}(\Omega)$ and $\mathrm{Dis}(\Omega)$ are used at each iteration, their impact on the result may be significant. So, just this pair of operations are further considered from the point of view of the accuracy problem.

Let Ω_B be the tth iteration result of a composed CA, and $\Omega_R = \mathrm{Av}(\Omega_B)$ should be obtained to make next operation compatible. Then according to (19) Boolean states $(v, m) \in \Omega_B$ are replaced by real ones from the finite set of numbers $Q = \{0, 1/q, \ldots, 1\}$, where $q = |\mathrm{Av}(m)|$. Hence, the error $E_{\mathrm{Av}}(m)$ incorporated by approximating a Boolean representation of a spatial function by discrete values from a finite set Q is constrained by

$$E_{\mathrm{Av}} \le \frac{1}{|\mathrm{Av}(m)|} = \frac{1}{q}. \tag{63}$$

Boolean discretisation of $\Omega_R = \{(u, m)\}$ performed according to (20) and resulting in $\Omega_B = \{(v, m)\}$ also brings in some errors. Probabilistic formula (20) provides that the obtained Ω_B in its averaged form is equal to the averaged state value $\langle v(m)\rangle \in Av(\Omega_B)$, which yields the following condition of the discretization accuracy.

$$\Omega_R = \mathrm{Av}(\Omega_B), \quad u(m) = \langle v(m)\rangle \quad \forall m \in M, \tag{64}$$

discretization error $E_{\mathrm{Dis}}(m)$ being the difference

$$E_{\mathrm{Dis}}(m) = |u(m) - \langle v(m)\rangle|. \tag{65}$$

The error vanishes in those cells where

$$u(m) = \langle v(m)\rangle = \frac{1}{q}\sum_{k=0}^{q-1} v(\phi_k(m)), \tag{66}$$

which happens very rarely, for example, when a fragments of a linear function or a parabola of odd degree is discretized. The error is most serious at the cells where $u(m)$ has extremes.

The most correct representation of discretization error is a function $E_{\mathrm{Dis}}(m, t)$, which shows possible deviations of $u(m, t)$ in all cells during the evolution. But, sometimes in the particular cases error values in a certain part of Ω, or maximal error in extremes of the spatial function at a certain time is of interest. For a general assessment of CA composition the mean discretization error at a given $t = \hat{t}$

$$E_{\mathrm{Dis}}(\hat{t}) = \frac{1}{|M|}\sum_{m\in M} |u(m, \hat{t}) - \langle v(m, \hat{t}, \rangle|, \tag{67}$$

is also used.

From (66) and (67) it follows that discretization errors depend on the averaging area size $q = |\mathrm{Av}(m)|$ and on the smoothness of $u(m)$ on $T_{\mathrm{Av}}(m)$. Both these parameters are conditioned by the discretization step h, which should be taken small, allowing q to be chosen large enough to smooth the extremes. Since $h = \mathbf{S}/|M|$, where $\mathbf{S}$ is a physical space under simulation, small h means large cellular array size.

The following experiment gives a quantitative insight to the accuracy problem.

Example 9. A half–wave of a sinusoid $u = \sin x,\ 0 < x < \pi$, is chosen for experimental assessment of discretization error dependence of $E_{\mathrm{Dis}}(\hat{t})$ on $|M|$ and on $\mathrm{Av}(m)$. The cellular array representation of a given continuous function is as follows

$$\Omega = \{(u(m), m)\}, \quad u(m) = sin\left(\frac{\pi}{|M|}m\right), \quad m = 0, 1, 2, \ldots, |M|. \qquad (68)$$

To obtain the dependence $E_{\mathrm{Dis}}(|M|)$, 30 discretizations $\{\mathrm{Dis}_k(\Omega) : k = 1, 2, \ldots, 30\}$ of the function given as (68) have been obtained with $|M_k| = 60 \times k$, which corresponds to the argument domain of the cellular array equal to $60 < |M_k| < 1800$, or to the sinus' argument domain in angular form equal to $2^\circ > h > 0.1^\circ$. Each $\mathrm{Dis}_k(\Omega)$ has been averaged with $|\mathrm{Av}_k(m)| = 0.2|M_k|$, and the mean errors $E_{Dis}(|M_k|)$ have been computed according to (67) (Fig. 13).

To obtain the dependence $E_{\mathrm{Dis}}(q)$, for the same Ω given as (68) 30 discretizations $\{\mathrm{Dis}_j(\Omega) : j = 1, 2, \ldots, 30\}$ have been obtained with fixed $|M| = 360$ but different $q_j = |\mathrm{Av}_j|$, where $q_j = 5 \times j$. Each $\mathrm{Dis}_j(\Omega)$ has been averaged with $\mathrm{Av}_j(m)$, and the mean errors $E_{\mathrm{Dis}}(q_j)$ have been computed according to (67) (Fig.14).

From Figs. 13 and 14 it may be concluded that

1) the mean error $E_{\mathrm{Dis}}(|m|)$ decreases with the increase of $|M|$ and does not exceed 1% with $|M| > 360$ which correspond to $h < 0.5^\circ$;

2) the mean error $E_2(|\mathrm{Av}(m)|)$ has a minimum when $|\mathrm{Av}(m)| \approx 36^\circ$, i.e. $|\mathrm{Av}(m)|_{E_{\mathrm{Dis}}=min} = 0.2|M|$.

From this example it follows, that regulating the smoothness of the extremes by appropriate choice of the CA size, the needed accuracy of CA composition may be achieved. Of course, the complexity of simulation increases linearly with the increase of $|M|$.

5.2 CA Composition Stability

There are two aspects of stability regarding CA composition. The first aspect concerns *behavioral stability,* which determines whether the CA evolution tends to a stable state or to a periodic cycling. The property is studied for simple Boolean CA–models in [2], but no method is known to check behavioral stability for an arbitrary CA. As for CA with real alphabets and nonlinear functions, their behavioral stability is a subject of nonlinear dynamic system

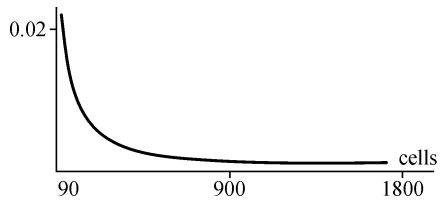


Fig. 13. Mean discretization error dependence on the naming set size $|M|$ with $|Av(m)| = 0.2|M|$ for cellular array (67)

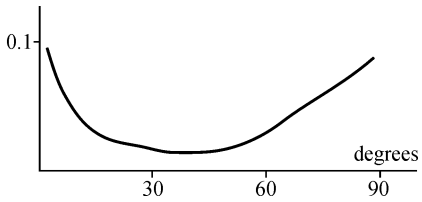


Fig. 14. Mean discretization error dependence on the naming set size of averaging area $|Av(m)|$ with $|M| = 360$ for cellular array (67)

theory and may be checked using its methods, as it is usually done in continuous mathematics (see for example, [35]). The problem which is of interest in connection with CA composition is as follows: all component CA being stable, is the composition obtained by the above techniques also stable? Till now the problem remains opened. Moreover, the current level of knowledge about the CA behavior stability does not seem to be high enough for the solution.

The second stability aspect is *computational stability*. This property is associated with the round-off errors, which are inevitable when float point arithmetics is used. This aspect of stability is more effectual for study because there are at least two particular cases of composition methods for which computational stability may be quantitatively assessed.

The first case comprises local and global sequential composition of Boolean CA–models. Since all alphabets are Boolean, there is no round-off errors, and since cellular arrays under processing have finite size, the resulting averaged values are bounded and stable.

The second case includes sequential or parallel global composition techniques of Boolean and real CA–models, where cellular array transformations $\mathrm{Av}(\Omega_R)$ and $\mathrm{Dis}(\Omega_B)$ are used at each iteration. In this case the following assertion is true: if $\aleph_R$ is stable, and, hence, its state values may be made bounded by the real closed interval $[0, 1]$, then the composition is computationally stable. This assertion is evident, at the same time it is of considerable importance for widely used diffusion–reaction processes, because it asserts, that composition of Boolean diffusion and real reaction is free of the so called Courant constraint imposed on the PDE counterpart of the process. The Courant constraint in PDE explicit solution is associated with second order partial derivatives of spatial coordinates (Laplace operator), representing a diffusive part in the PDE. For example, for 2D case, it forbids the value $C_{PDE} = \frac{\tau d}{h^2}$ to exceed 0.25, where h is a space step, d is a diffusion coefficient. From the above it follows that the time step $\tau < \frac{1}{2}\frac{h^2}{d}$, should be small enough, which results in significant increase of computation time. Meanwhile, simulating the diffusion part by a CA, no care should be taken about the stability, the constraint being imposed by the dimensionless diffusion coefficient, which is the characteristic of a CA-model. For example, a naive CA-diffusion [3] has $C_{CA} = 0.5$, In the CA-model proposed in [26] C_{CA} depends on averaging radius r, bein equal to $C_{CA} = r(r + 1)/6$ for 2D case.

Example 10 . A diffusion–reaction process called a propagating front is simulated by two models: (1) explicit finite-difference method of PDE solution and (2) composition of a Boolean diffusion CA and a real reaction CA. The PDE is as follows. The process is initiated by a dense square 80×80 of propagating substance in the center of an area 639×639. The border conditions are periodic.

$$\frac{\partial u}{\partial t} = d\left(\frac{\partial u^2}{\partial x^2} + \frac{\partial u^2}{\partial y^2}\right) + 0.5u(1 - u), \tag{69}$$

where $d = 0,33$ cm^2/s. The discretized 2D space is $M = \{(i,j) : i,j = 0,1,\ldots,i,\ldots,639\}$. Initial state for PDE solution is

$$u^{(0)}(i,j) = \begin{cases} 1, \text{ if } 280 < i,j < 360, \\ 0, \text{ otherwise,} \end{cases}$$

The finite difference representation of the diffusion part of (69) is as follows

$$u^{(t+1)}(i,j) = u^{(t)}(i,j) + 0.33(u^{(t)}(i-1,j) + u^{(t)}(i+1,j) + u^{(t)}(i,j+1) + u^{(t)}(i,j-1) - 4u^{(t)}(i,j)). \tag{70}$$

With the time-step $\tau = 1$ s and the space step $h = 1$ cm, the Courant value $C_{PDE} = td/h^2 = 0.33$, which is out of Courant constraint. So, the function $u^{(t)}(i,j)$ obtained by (70) is not stable.

The same process may be simulated by a superposition $\aleph = \Psi(\aleph_{\text{diff}}, \aleph_{\text{reac}})$. The first component $\aleph_{\text{diff}} = \langle A, M, \hat{M}, \theta_{\text{diff}}, \sigma \rangle$ is in its turn a superposition of the two-stage synchronous CA from Example 1 (Sec.3.1) with θ given as (36), and the operator of averaging, i.e.

$$\theta_{\text{diff}}(i,j) = \text{Av}(\theta(i,j)), \tag{71}$$

resulting in a global configuration $\Omega_{\text{diff}}(t) = \{(u,(i,j)) : u \in [0,1]\}$

The second component $\aleph_{\text{reac}} = \langle A, M, \hat{M}, \theta_{\text{reac}}, \sigma \rangle$ is a context-free CA computing in each cell a reaction function $f(u) = 0.5u(1-u)$ with subsequent discretization, where $A, M, \hat{M}, \sigma$ are equal to those of $\aleph_{\text{diff}}$, the local operator being

$$\theta_{\text{reac}} : (u,(i,j)) \to (v,(i,j)), \quad v = \text{Dis}(f(u),(i,j)). \tag{72}$$

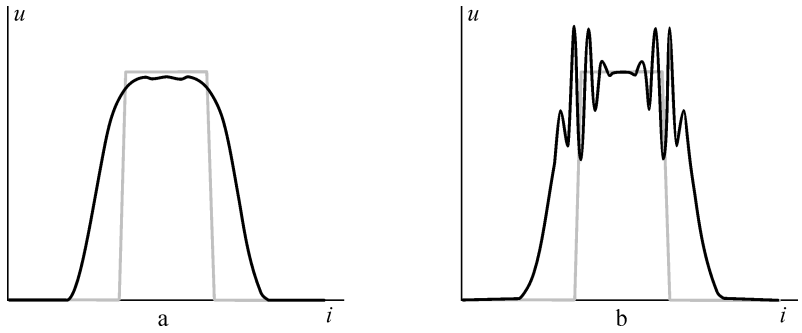


Fig. 15. Simulation of 2D propagation front initiated by a dense square in the central part of cellular space, a profile $u(i,319)$ is given obtained by : (**a**) CA superposition of $\aleph_{\text{diff}}$ with θ_1 (71) and $\aleph_{\text{reac}}$ with θ_2 (72), (**b**) solution of finite-difference equation (70)

.

Snapshots of both processes (PDE solution) and (CA superposition) after 20 iterations are shown in Fig.15. It is seen, that evolution of CA superposition is absolutely stable, while finite-difference solution of (71) exhibits a divergence.

5.3 Composition Complexity

Here, an attempt is made to assess how much of additional work a composed CA has to do, as compared with the total complexity of the components. Such an assessment cannot be precisely done. There are many reasons for that. The most significant are the following: (1) complexity relations between different arithmetic operations strongly depend on hardware architecture; (2) the same is true for comparing Boolean and real operations; (3) complexity of performing CA transition functions range from $O(n)$ to $O(2^n)$, n being the cardinality of the neighborhood in the local operator. Nonetheless, an insight may be given on the relation between the complexity of transition function computation and that of transformations needed for providing compatibility.

In case when sequential local asynchronous composition and global synchronous composition techniques contain no averaging and discretization operations, no additional time is needed and the total number of elementary operations is equal to the sum of those of the component CA.

When global synchronous composition, no matter sequential or parallel, is used, transformations of Boolean cellular array into a real one and vice versa are to be performed at each iteration. In such a case, iteration time is increased by t_{add} additional elementary operations.

$$t_{\text{add}} = |M| \times (t_{\text{Av}} + t_{\text{Dis}}), \tag{73}$$

where t_{Av} and t_{Dis} are numbers of elementary operations which have to be executed by a cell while performing averaging according to (19), or discretization according to (20), respectively. As for t_{Av}, it is clearly seen from (19), that the time needed to compute $\text{Av}(v_m)$ may be assessed as $t_{\text{Av}} = C_{\text{Av}} \times |\text{Av}(m)| \times \tau$ where $C_{\text{Av}} \approx 1$ is a constant, τ — the time of elementary function execution. The discrertization time $t_{\text{Dis}} = C_{rand}$, so, according to (20) it depends only on the random number generator time, which may be taken $C_{rand} < 5$. Since the transformation is used in the composition techniques where both Boolean and real components are included, the time t_{add} should be compared with Boolean and real transition functions computation time $t_{\text{comp}} = t_B + t_R$, where $t_B = C_B \times \tau$ and $t_R = C_R \times \tau$. The coefficients C_B and C_R essentially depend on the character of the transition functions but, usually, both functions require to execute not more than 100 elementary operations.

Comparison of t_{add} with t_{comp} yields:

$$\frac{t_{\text{add}}}{t_{\text{comp}}} = \frac{C_{Av} + C_{Dis}}{C_B + C_R},$$

which enables us to conclude that t_{add} and t_{comp} have identical order of complexity, hence, Boolean–real transformations increase the computation time about twice.

6 Conclusion

Till now, no mathematical method and no promising approach is known to CA synthesis from a given description of its evolution. Nevertheless, some way out should be found. A simple one is to follow a well known approach used in PDE theory which implies composing PDE systems out of a set of differential operators and functions. Such an approach seems to be expedient when considering the following similarities between CA composition and PDE system construction. For example, first order and second order differential operators in PDEs over the space have their CA counterparts in the form of shift and diffusion local operators, respectively. And in both cases for obtaining a mathematical model of reaction–diffusion process those operators are composed with nonlinear reaction functions. Unfortunately, the above similarities are only particular cases. In general, there is no formal procedure to obtain a CA simulating space–time nonlinear behavior. It is just the fact that has provoked the development of compatible algebraic operations on cellular arrays, allowing to integrate continuous functions into a CA composition techniques.

But the most important destination of CA composition is not in presenting another way of simulating processes which may be described in terms of PDE, but in obtaining capability of constructing mathematical models for those phenomena, for whom no other mathematical description is known. Such a processes are mostly associated with the fields of science which are in the initial stage of development. For example, plant growth mechanisms, embryo fetation, cellular division, morphogenesis – from biology; surface oxidation, chemical reaction on catalyst, dissociation, adsorption – from chemistry; epitaxial growth, crack formation, rubber deformation, robotics – from engineering; tumor growth – from medicine, etc. Of course, the available experience in science and engineering is not sufficient to forecast the future of CA simulation methodology. Anyway, now it is clear that only a small part of the huge amount of CA–models have evolutions which resemble natural phenomena, and, hence, may be used for simulation. Moreover, those, which occur to be helpful, ought to be enriched by some additional properties, such as probability in transition functions, complicated modes of operations, composite alphabet, non-homogeneous cellular space, etc.. All these, being oriented to obtain CA–models of complex phenomena, require a unique formalism for composing complex CA–models from a number of more simple ones. The above considerations allow to hope that the presented attempt to construct a systematic approach to CA composition is not futile.

References

1. Toffolli T, Margolus N (1987) Cellular Automata Machines. MIT Press, USA. Physica D 10: 117–127
2. Wolfram S. (2002) A new kind of science. Wolfram Media Inc., Champaign, Ill., USA.
3. Bandman O (1999) Comparative Study of Cellular Automata Diffusion Models. In: Malyshkin V (ed) PaCT-1999. Springer, Berlin. LNCS 1662: 395–404
4. Park J K, Steiglitz K, Thurston W P (1986) Physica D 19: 423-432
5. Vannozzi C, Fiorentino D, D'Amore M, Rumshitzki D S, Dress A, and Mauri R (2006) Indust Eng Chem Res 45(4): 2892–2896
6. Creutz M (1996) Celllular Automata and Self-organized Criticality. In: Bannot G, Seiden P (eds) Some new directions in science on computers. World Scientific, Singapore 147–169
7. Rothman D H, Zalesky S (1997) Lattice-Gas Cellular Atomata. Simple Model of Complex Hydrodynamics. Cambridge University press UK
8. Succi S (2001) The Lattice Boltzmann Equation for Fluid Dynamics and Beyond. Oxford University Press, New York
9. Axner L, Hoekstra A G, Sloot P M A (2007) Phys Rev E 75, 036709
10. Malinetski G G, Stepantsov M E (1998) Zhurnal Vychislitelnoy Matematiki i Mathematicheskoy Phisiki 36 (6): 1017–1021
11. Frish U, d'Humieres D, Hasslacher B, Lallemand P, Pomeau Y, Rivet J P (1987) Complex Systems 1: 649-707
12. Elokhin V I, Latkin E I, Matveev A V, Gorodetskii V V (2003) Kinetics and Catalysis 44 (5): 672–700
13. Jansen A P J (2003) An Introduction to Monte-Carlo Simulation of Surface Reactions. arXiv: cond-mat/0303028 v1 3
14. Neizvestny I G, Shwartz N L, Yanovitskaya Z Sh, Zverev A V (2002) Comp Phys Comm 147: 272–275
15. Saum M A, Gavrilets S. (2006) CA Simulation of Biological Evolution in Genetic Hyperspace. In: El Yacoubi S, Chopard B, Bandini S (eds) ACRI-2006. Springer, Berlin, LNCS 4176: 3–13
16. Wu Y, Chen N, Rissler M, Jiang Y, Kaiser D, Alber M (2006) CA Models of Myxobacteria Sworming. In: El Yacoubi S, Chopard B, Bandini S (eds) ACRI-2006. Springer, Berlin. LNSC 4176: 192–203
17. Ghaemi M, Shahrokhi A (2006) Combination of the Cellular Potts Model and Lattice Gas Cellular Automata for Simulating the Avascular Cancer Growth. In: El Yacoubi S, Chopard B, Bandini S (eds) ACRI-2006. Springer, Berlin. LNSC 4176: 297-303
18. Slimi R, El Yacoubi S (2006) Spreadable Probabilistic Cellular Automata Models. In: El Yacoubi S, Chopard B, Bandini S (eds) ACRI-2006. Springer, Berlin. LNSC 4176: 330-336
19. Bandini S, Manzoni S, Vizzari G (2004) IEICE Trans on Inf Syst E87-D: 669-676
20. Adamatsky A (2005) Dynamics of Crowd-Minds. In: Series on Nonlinear Science 54 World Scientific
21. Biondini F, Bontempi F, Frangopol D M, Malerba P G (2004) J Struct Eng 130 (11): 1724–1737
22. Bandman O (2005) Composing Fine-Grained Parallel Algorithms for Spatial Dynamics Simulation. In: Malyshkin (ed) PaCT-2005. Springer, Berlin. LNCS 3606: 99–113

23. Wolfram S (1984) Physica D 10: 1-35
24. Chua L O (2002) CNN: a Paradigm for Complexity. World Scientific, Singapore
25. Achasova S, Bandman O, Markova V, Piskunov S (1994) Parallel Substitution Algorithm. Theory and Application. World Scientific, Singapore.
26. Weimar L R, Boon J-P (1994) Phys Rev E 49: 1749-1752
27. Bandman O (2002) Simulating Spatial Dynamics by Probabilistic Cellular Automata. In: Bandini S, Chopard B, Tomassini M (eds) ACRI-2002. Springer, Berlin. LNCS 2493: 10–20.
28. Bandman O (2003) Bulletin of Novosibirsk Computer Center series Computer Science 19: 10-19.
29. Bandman O (2007) Coarse-Grained Parallelization of Cellular-Automata Simulation Algorithms. In: Malyshkin V (ed) PaCT-2007. Springer, Berlin. LNCS 4671: 370–384
30. Ziff R M, Gulari E, Barshad Y (1986) Phys. Rev. Lett. 56: 2553
31. Vichniac G (1984) Physica D 10: 86–112.
32. Svirezhev Yu(1987) Nonlinear Waves, Dissipative Structures and Catastrophes in Ecology. Nauka, Moscow
33. Schlogl F (1972) Zh.Physik 253: 147–161
34. Schrenk C P, Schutz P, Bode M, Purwins H-G (1998) Phys Rev E 5 (6): 6481–5486
35. Michel A N, Wang K, Hu B (2001) Qualitative Theory of Dynamics Systems: The Role of Stability Preserving. New York, CRC Press.

Index