Using Multi Core Computers for Implementing Cellular Automata Systems *

Olga Bandman

Supercomputer Software Department
ICM&MG, Siberian Branch, Russian Academy of Sciences
Pr. Lavrentieva, 6, Novosibirsk, 630090, Russia
E-mail: bandman@ssd.sscc.ru

Abstract. A concept of cellular automata system (CA-system) is introduced as a model of comp[lex phenomena in which several interacting species are involved. CA system suggests a common work of several CA where each processes its own cellular array using in its transition rules cell states of others CA of the system. Taking into account that multi core computers with shared memory are nowadays widely used, a temptation to accelerate the computation by allocating each CA of the system onto one of computer cores is quite natural. Hence, it would be helpful to know what speedup can be obtained by such a parallelization. The paper is aimed to get an answer to this question by determining the conditions, when multicore parallel implementation of CA systems is expedient and correct, and develop the parallelization algorithms for typical CA systems. The results are illustrated by simulation experiments.

1 Introduction

Cellular Automata (CA) being regarded as a model of spatial dynamics, gradually changes its status of object of study for the status of the method for studying natural processes. CA properties such as nonlinearity of transition functions and irreversibility of evolution made them particularly useful for investigating the behavior of complex systems [1], exhibiting self organization and emergency. The number of such investigations increases rapidly, comprising the study of new more complicated phenomena in biology, physics and chemistry [2]. Many of such phenomena are simulated by parallel composition of several CA [3], the evolution of each CA component simulating the corresponding species behavior. Parallel CA composition is a set of CA, operating in common in such a way that each CA transition functions variables are cell states of any CA of the system. By analogy to partial differential equations of traditional numerical analysis, parallel composition of CA are further called *CA systems.* Nowadays CA systems are used mostly in scientific investigation being implemented on personal computers, many computational experiments on one and the same CA being performed

* Supported by (1) Presidium of Russian Academy of Sciences, Basic Research Program N 2 (2009), (2) Siberian Branch of Russian Academy of Sciences, SBRAS Interdisciplinary Project 32 (2009), (3) Project RFBR 11-01-00567a.

with many different parameters, which should be easily and promptly changed. Although the computational time is wanted to be as small as possible, there is no need to run the programs on remote powerful clusters, each time waiting for the results. But, having a two-, four- or eight-core computer with a shared memory on the table, it is reasonable to make the cores operate in parallel, in correspondence with the parallel composition of CA in the system. Whether it is worth to be done and what are the conditions for such a parallelization be efficient, is the subject of the paper.

The paper is organized as follows. Next section presents the method of parallel composition which provides correctness conditions conservation. In the third section parallel algorithms for two-core implementation of a single reaction-diffusion process simulation is given. Fourth section is devoted to parallel multi-core implementation of a CA system, simulating several interacting reaction-diffusion processes. In the Conclusion the results are summarized and application perspectives are outlined.

2 Formal Definition of a Cellular Automata System

In general case CA-system suggests any number n of CA working in common. But since formal definitions for arbitrary n are very cumbersome and hardly comprehensive, for clearness and without loss of generality, the system $\aleph = \Upsilon(\aleph_1, \aleph_2)$ with $n = 2$ is further considered. Each component $\aleph_k$, $k = 1, 2$, is determined by three sets, $\aleph_k = \langle A_k, X_k, \Theta_k \rangle$, where A_k is a state alphabet, X_k - a set of cell names (coordinates in finite discrete space), and Θ_k – a local operator. The alphabets A_1, A_2 may be different and of any type (Boolean, real, symbolic). Both synchronous and asynchronous modes of operation are allowed.

Between X_1 and X_2, $i = 1, 2 \ldots, I$, $I = |X|$, there exists an one-to one correspondence $\xi : X_1 \to X_2$:

$$\begin{array}{ll} x_2 = \xi(x_1), & \forall x_2 \in X_2, \\ x_1 = \xi^{-1}(x_2), & \forall x_1 \in X_1. \end{array} \tag{1}$$

The elementary entity of a CA is a *cell* represented by a pair (v, x), where $v \in A$, $x \in X$, the state of the cell x being denoted as $v(x)$ or v_x. The set $\Omega = \{(v, x)\}$ containing $|X|$ cells with different names forms a *cellular array.* A pair of cells in $\Omega_1 \cup \Omega_2$, such that $x_1 = \xi^{-1}(x_2)$ are further referred to as *twin-cells.* When a cell of any component CA is meant, it is named simply as x . Similarly, in all expressions valid for all CA components, the bottom indices are removed.

In X_1 and X_2 two types of templates $T(x)$ are defined as follows. Let $d(x_i, x_j)$ be a distance between x_i and x_j, $x_i, x_j \in X_k$. Then

$$T_{kk}(x_i) = \{x_j : d(x_i, x_j) < r,\ x_j \in X_k\} \tag{2}$$

represents a *base template* with radius r, and

$$T''_{kl}(x_i) = \{x_j : d(\xi(x_i), x_j) < r_l,\ x_i \in X_k,\ x_j \in X_l\},\ k \neq l,\ r_l << |X|, \tag{3}$$

is a *remote context* template with radius r_l.

The set of states

$$V(T(x)) = \{v(x_j) : x_j \in T(x)\} \tag{4}$$

form a *local configuration*, with underlying template $T(x)$. The set of states of all cells in $\Omega_k = \{(v, x_i) : \forall x_i \in X_k\}$ is referred to as *global configuration* $V(X_k) = \{v(x_i) : \forall x_i \in X_k\}$.

Each Θ_k is expressed by a substitution [4] as follows.

$$\Theta_k(x): \quad V(T_{kk}(x)) \star V^{(}T''_k(x)) \to V'(T_{kk}(x)), \quad k, l = 1, 2. \tag{5}$$

The template $T''_k(x)$ contains two parts, i.e,

$$T''_k(x) = T''_{kk}(x) \cup T''_{kl}(x), \tag{6}$$

where $T''_{kk}(x)) \subset X_k$ is a *self context* and $T''_{kl}(x)) \subset X_l$ is a *remote context*.

The local configuration $V(T_{kk}(x))$ in (5) is called a *base* of $\Theta_k(x)$. Its states are to be replaced by $V'(T_{kk}(x))$, when $\Theta_k(x)$ is applied. The local configuration $V(T''_k(x))$ is called a *context*. Its states are not changed by $\Theta_k(x)$ application, but serve as variables in the *transition functions* f_{kj}, whose values $v'(x_j)$ are states in the *next state local configuration* $V'(T_{kk}(x))$:

$$v'(x_j) = f_{kj}(V(T_{kk}(x) \cup T''_k(x)), \quad x_j \in T_{kk}. \tag{7}$$

An application of Θ_k to a cell $x \in X_k$ means substituting of states $v'(x_j) \in V'(T_{kk}(x))$ for $v(x_j) \in V(T_{kk}(x))$.

If a CA system is a parallel composition $\aleph = \Upsilon(\aleph_1, \aleph_2)$, the two cellular arrays are processed in parallel by application Θ_1 to Ω_1, and Θ_2 to Ω_2. The whole process of simulation consists of a sequence of *iterations*. An iteration presumes that in both CA the corresponding operator has been applied to all cells, which yields a *global transition* of the system: $\Omega_1(t) \cup \Omega_2(t) \to \Omega_1(t+1) \cup \Omega_2(t+1)$.

In a CA system any mode of operation is allowed: synchronous CA may interact with an asynchronous one, provided correctness conditions for both and for the system in a whole are satisfied, assuming each CA operates according to the following algorithms.

Synchronous CA performs a global transition from $\Omega(t)$ to $\Omega(t+1)$ as follows:

1) $\Omega(t)$ is copied to $\Omega'(t)$.

2) $\Theta(x)$ is applied to all cells from $\Omega'(t)$, next values $v'(x)$ being computed according to (7) and written in twin-cells of $\Omega(t)$.

Asynchronous CA performs a global transition by executing $|X|$ times the following steps:

1) A cell is randomly chosen from X.

2) The next state $v'_i(x)$ is computed by (7), and the substitution (5) is immediately performed.

3 Correctness of CA System Functioning

CA simulation is considered as an effective computation [5], if it possesses the following properties *safeness* and *fairness*. The first provides any datum not being lost during the simultaneous application of $\Theta_k(x_i)$ and $\Theta_k(x_j)$. A sufficient condition for that is formally expressed as follows:

$$\big(T_{kk}(x_i) \cup T''(x_i)\big) \cap T_{kk}(x_j) = \emptyset \quad \forall x_i, x_j \in X_k. \tag{8}$$

The second property guarantees that all cells of X_k have equal rights to be chosen for Θ_k application. Synchronous CA, functioning according to the algorithm (sec.2), satisfy safeness condition, if it has a single-cell base, i.e. if $|T_{kk}| = 1$. If it is not so, the CA should be transformed into a superposition of $|T_{kk}|$ single-cell base CA according to a method given in [4]. At any case, the next states of $\Theta_k(x_i)$ application to Ω_k should be written to an additional array Ω'_k which is intended for storing the remote contexts. This prevents to change cell states in the left-hand side of (8) that have not been used by application of $\Theta_k(x_j)$, $x_i \neq x_j$. Fairness is satisfied since all cells in X are chosen for being processed with equal probability.

As distinct to synchronous case, in asynchronous CA transition functions (7) are allowed to be applied both to current and to next states and, hence, multi-cell base in local operators are frequently used. So, to guarantee safeness the only requirement is that local operator $\Theta_k(x)$ is to be indivisible, i.e. nothing is allowed to occur between changing the states of $V'(x)$ during its execution. This fact makes useless the additional array when CA operates alone. Fairness is guaranteed by using equal random distribution when choosing cells for Θ application.

Besides the requirement of all system CA correctness, there are some additional conditions to be met for the whole system functioning be safe and fair.

1) Application of Θ_k to any cell of Ω_l, $k \neq l$, should be forbidden, otherwise a state might be lost for being used by another application.

2) During the application of $\Theta_k(x)$ to Ω_k,when states from $V(T_{kk}(x))$ are sequentially changing, no cell of $T_{kk}(x)$ shouldbe used as remote context for any Θ_l. Otherwise some states of Θ''_l might be lost.

Formally the above two statements are expressed as follows.

$$\big(T_{kk}(x_i) \cup T''(x_i)\big) \cap T_{ll}(x_j) = \emptyset \quad \forall x_i, x_j \in X_k \cup X_l, \quad k, l = 1, 2. \tag{9}$$

In a more comprehensive form, the correctness conditions of CA-system may be formulated as follows:

1) All CA of the system should be correct, i.e. satisfy (8).

2) If states of Ω_k serve as a remote context to Θ_l, then there should be a copy of $\Omega_k(t)$, referred to as $\Omega'_k(t)$, whose cell states are included in remote context local configurations $V(T''_{lk})$, in order to satisfy (9).

If in a CA system there is such a CA, say $\aleph_k$, whose cell states are not used in a remote context of any other CA in the system, then $\aleph_k$ operates according to its mode as if it were alone. In Fig.1 two variants of CA interaction in the system

are schematically shown. Moreover, in the case when $\aleph_1$ operates independently and $\aleph_2$ is asynchronous, Ω_2' is redundant.

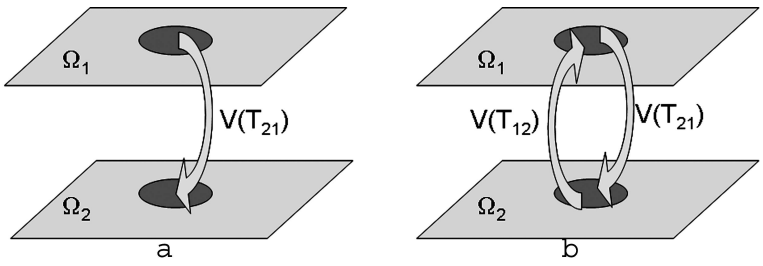

Fig. 1. Schematic image of a two types of CA systems: a) when one CA operates independently, b) when both CA are mutually dependent

4 CA-System Simulating a Single Reaction-Diffusion Process

A wide class of phenomena exhibiting complex behavior are dynamical systems where several species move and interact chemically, physically or biologically. Capability of moving is usually independent on reactive interactions, the latter being associated with dissipative (mostly, chemical) character of processes under simulation. Hence, the behavior of each species is described by a pair of typical CA, which are referred to as *diffusion CA* and *reaction CA*. Such a pair in its turn forms a simple diffusion-reaction CA-system, which usually constitutes a *building block* for construction of *complex CA-systems*, simulating several interacting reaction-diffusion processes.

Let $\aleph_1$ and $\aleph_2$ form a reaction-diffusion block simulating diffusion and reaction, respectively (Fig.1a). Parallel algorithm for allocating the system to a multicore processor is as follows.

- Create initial cellular arrays $\Omega_1(0)$ and $\Omega_2(0)$.
- For each iteration $t = 1, \ldots, T$:
  begin parallel computation
  - thread 1
    copy $\Omega_1(t)$ to $\Omega_1'(t)$
    apply Θ_1 to $\Omega_1(t)$
  - thread 2
    copy $\Omega_2(t)$ to $\Omega_2'(t)$
    apply Θ_2 to $\Omega_2(t)$ reading the remote context $V_{T_{21}}$ from $\Omega_1'(t)$.

  end parallel computation
- Copy the resulting Ω_2 to Ω_1, $t \to t + 1$.

It is worth to be noted that the above algorithm is valid for both synchronous and asynchronous diffusion CA.

Example 1. A system of two interacting CA is used to simulate a pattern formation process [7] induced by a chemical reaction on a metallic surface, the latter having been heated in its central part. The CA $\aleph_v = \langle A_v, X_v, \Theta_v \rangle$ simulates the propagation of heat over the surface, being the asynchronous well known diffusion CA called a *naive diffusion* [6].
$\aleph_u = \langle A_u, X_u, \Theta_u \rangle$ simulates the process of patterns emergency on the surface. It is a synchronous CA of majority type with weighted template. The influence of changing temperature on pattern formation process is reflected by the dependence of weighted template entries on the averaged twin cells states of $\aleph_v$.

Fig. 2. Initial state of the cellular array $\Omega_v(0)$ from Example 1

Both CA have Boolean alphabet $A = \{0, 1\}$, their naming sets satisfying (1), $|X_v| = |X_u| = \{(i, j) : i, j = 0, \ldots, 300\}$.

CA $\aleph_v$ operates independently simulating the propagation of heat over the whole area, the initial distribution of temperature is shown in (Fig.2). According to naive CA-diffusion its local operator performs the exchange of states between a cell $((v_0, (i, j)_v)$ and one of its four neighbors $(v_k, (i, j)_v)$, $k = 1, 2, 3, 4$, using von Neumann template

$$T_v = \{(i, j)_v\}, \quad T''_{vv} = \{(i, j + 1)_v, (i - 1, j)_v, (i, j - 1)_v, (i + 1, j)_v\}, \tag{10}$$

and a transition function

$$\begin{aligned} v'_0 &= v_k, \quad \text{if} \quad 0.25k < rand < 0.25(k + 1), \\ v'_k &= \begin{cases} v_0 & \text{if} \quad 0.25k < rand < 0.25(k + 1), \\ v_k & \text{otherwise.} \end{cases} \qquad k = 1, \ldots, 4. \end{aligned} \tag{11}$$

Local operator $\aleph_u$, operating in $\Omega_u = \{(u, (i, j)_u : (i, j) \in X_u\}$ has a single cell base, the context template including cells both from X_v and X_u.

$$T''_{uv} = \{(i + g, j + h)_u, (i + g, j + h)_v : g, h = -3, \ldots, 3\}, \tag{12}$$

the transition function being as follows:

$$u'_0 = \begin{cases} 1, & \text{if} \quad \sum_{g=-r}^{r} \sum_{h=-r}^{r} w_{gh} u_{i+g,j+h} > 0.1, \\ 0, & \text{otherwise,} \end{cases} \tag{13}$$

where

$$w_{gh} = \begin{cases} 1, & \text{if} \quad |g| \leq 1 \ \& \ \ |h| \leq 1, \\ -\langle v_{i+g,j+h} \rangle & \text{otherwise.} \end{cases}$$

In Fig.3 three snapshots of pattern formation process in Ω_u are shown.

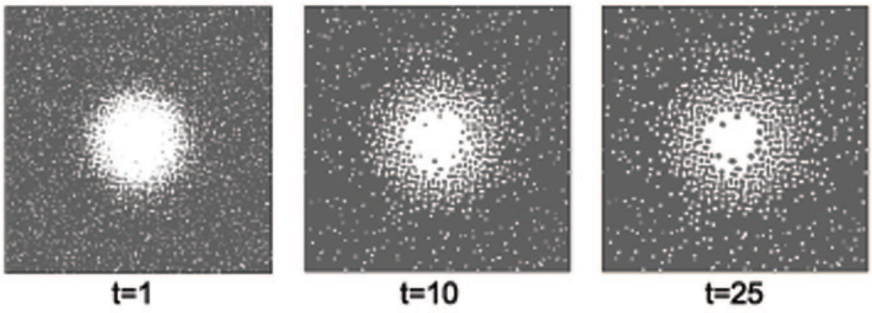

Fig. 3. Three snapshots of the evolution of a pattern formation CA $\aleph_u$ in a CA system where inhibitor values are controlled by the heat propagation $\aleph_v$ with initial state $\Omega_v(0)$ shown in Fig.2

The same process has been simulated by using diffusion CA $\aleph'_v$ of synchronous type with Margolus neighborhood [6]. The local operator of the latter is a superposition of two ones, $\Theta'_v = \Theta_1(\Theta_2)$ both being contextless, each having four cells in its base.

$$\begin{aligned}\Theta_1(i,j)_v = &\{((v_0,(i,j)_v),(v_1(i,j+1)_v),(v_2(i+1,j+1)_v),(v_3,(i+1,j)_v)\} \rightarrow \\ &\{((v'_0,(i,j)_v),(v'_1,i,j+1)_v),(v'_2(i+1,j+1)_v),(v'_3,(i+1,j)_v)\}, \\ \Theta_2(i,j)_v = &\{((z_0,(i,j)_v),(z_1(i,j+1)_v),(z_2(i+1,j+1)_v),(z_3,(i+1,j)_v)\} \rightarrow \\ &\{((z'_0,(i,j)_v),(z'_1,i,j+1)_v),(z'_2(i+1,j+1)_v),(z'_3,(i+1,j)_v)\},\end{aligned} \tag{14}$$

where

$$v'_k = v_{(k+1)mod4}, \quad z'_k = z_{(k-1)mod4}. \tag{15}$$

For both above CA systems, coordination of heat propagation rate and that of pattern formation is achieved by forming each iteration of the system by including in it Dv iterations of diffusion , and one iteration of pattern formation. The value of Dv may vary from $Dv = 10$ to $Dv = 1000$. This fact gives us the opportunity to test efficiency of parallel implementation with different load balance between threads, by changing the value of Dv.

Computational experiments have been performed on the computer `Intel core i7` for two types of reaction-diffusion CA systems: 1) with asynchronous naive diffusion CA and 2) with Margolus CA-diffusion of synchronous type for five different values of Dv in each case. The results of parallelization efficiency are given in Fig.4 in the form of speedup dependence on the ratio $\rho = T(diff)/T(react)$, where $T(diff)$ and $T(react)$ are diffusion and pattern formation one iteration computation times, respectively. It is seen that for both synchronous and asynchronous case the efficiency is perfect when $\rho = 1$. Also, it is high enough ($> 0,7$) with $0.7 < \rho < 1.3$.

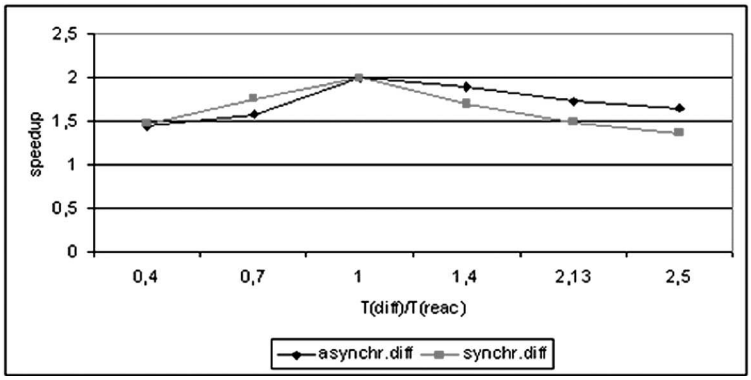

Fig. 4. Dependence of parallel two thread implementation efficiency on the ratio $\rho = T(diff)/T(react)$

5 CA System Simulating Many Interacting Processes

When many CA are functioning in common their interactions may be configured differently: some of them may operate independently, others may be interdependent. Correctness condition for any system is also expressed by (8) for each component CA and by (9) with the account that $T''(x_i)$ may be the union of several remote context templates. The most frequently studied is the type of CA system consisting of several reaction-diffusion blocks, for which two parallel implementation being possible:

1) each block is implemented as a single thread, reaction and diffusion CA running sequentially,

2) in each block reaction and diffusion are implemented as two parallel threads, hence, a system of n interacting blocks requires $2n$ threads.

Of course, any intermediate case is possible, the best variant being when the computation load in all threads is close to be identical.

A typical example for testing multithread implementations of a complex reaction-diffusion system is a prey-predatory problem which is a well known one in mathematical ecology [8].

Example 2. Prey-predatory problem [8] is usually represented by a system of two PDEs.

$$\begin{aligned} u_{tt} &= d_u u_{xx} + F_u(u,v), \\ v_{tt} &= d_v v_{xx} + F_v(u,v), \end{aligned} \tag{16}$$

where d_u, d_v are diffusion coefficients for two species, functions $F_u(u,v)$ and $F_v(u,v)$ are usually given in the form of polynomials of both variables. Let us interpret the problem in such a way: some predator (fish, deers) eat prey (plankton, moss). If there is enough of food, predator density increases (predator propagates) with the probability depending on satiated predator density. In case of food shortage predator density diminishes (predator die of hunger). Prey always

attempts to propagate, when not being eaten dy the predator. Since predator is usually more agile than prey, its diffusion is essential, as for prey diffusion – it is hardly observable, ($d_v >> d_u$).

CA system, representing prey-predatory interaction consists of two reaction-diffusion blocks: Υ_u and Υ_v, each being a simple systems of two automata $\Upsilon_u = \langle \aleph_{u1}, \aleph_{u2} \rangle$, $\Upsilon_v = \langle \aleph_{v1}, \aleph_{v2} \rangle$, in both blocks the first CA simulates diffusion, the second – the reaction. Correspondingly, $\aleph_{u1} = \langle A_{u1}, X_{u1}, \Theta_{u1} \rangle$, $\aleph_{v1} = \langle A_{v1}, X_{v1}, \Theta_{v1} \rangle$.

All CA have Boolean alphabets. Any pair of $X_{lk} = \{(ij)_{lk}\}$, $(l = u, v)$, $k = (1, 2)$. meets the relation (1). In both blocks diffusion simulation is performed by using synchronous CA with local operators Θ_{v1} and Θ_{u1} given in Example 1 by (14), differing only in diffusion coefficients, which are in correspondence of prey and predator agility, expressed in the model by the number of iterations $D_v = 50$ or $D_u = 1$ to be performed during one iteration of the CA system.

Local operators of reaction CA Θ_{v2} and Θ_{u2} represent the behavior of predator and prey and depend on both local densities

$$V((i,j)_v) = \frac{1}{|Av(i,j)_v|} \sum_{(k,l) \in Av(i,j)_v} v(k,l)_v, \tag{17}$$
$$U((i,j)_u) = \frac{1}{|Av(i,j)_u|} \sum_{(k,l) \in Av(i,j)_u} u(k,l)_u,$$

where

$$Av(i,j) = \{(k,l) : \ k = i + g, l = j + h, \ \ g, j = -r, \ldots, r\},$$

$r = 8$ being the radius of averaging in both cellular arrays. For the predator local operator is as follows:

$$\Theta_{v2} : \ \{v(i,j)_{v2}\} \star V(T''((i,j)_{v2}) \rightarrow \{v'(i,j)_{v2}\} \tag{18}$$

where

$$T''(v(i,j)_{v2}) = \{(k,l)_{v1}, (k,l)_{u1} : k = (i+g)_{v1}, l = (j+h)_{v1} \\ g, j = -r, \ldots, r\} \tag{19}$$

and the next state value

$$v'((i,j)_{v2}) = \begin{cases} 0, & \text{if } \ \Delta V(i,j) < 0 \quad \& \quad (rand) < p_{v \rightarrow 0}, \\ 1, & \text{if } \ \Delta V(i,j) > 0 \quad \& \quad (rand) < p_{v \rightarrow 1}, \end{cases} \tag{20}$$

where

$$\Delta V(i,j) = V((i,j)_v) - U((i,j)_u)$$

.

The probabilities $p_{v \rightarrow 0}$ and $p_{v \rightarrow 1}$ are determined based on the following considerations:

- If $\Delta V(i,j) > 0$, predator has lack of food and may die. So, $\Delta V((i,j)$ cell states in $Av((i,j)_{v2})$ should be inverted into "zero", yielding in $p_{v \rightarrow 0}$ be equal to the ratio $\Delta V(i,j)/U(i,j)$ [10].

- If $U(i,j) > V(i,j)$, $\Delta V((i,j)_{v2} < 0$, then predator has plenty of food at the place and propagates increasing its density according to the propagation function of the form $F_u(V) = cV(i,j)(1 - V(i,j)$, where $c = 0.5$ is a coefficient corresponding to the type of predator.

So, the probabilities in (20) are

$$\begin{array}{lll} p_{v\to 0} = \Delta V(i,j)/U(i,j), & \text{if} & \Delta V(i,j) > 0 \\ p_{v\to 1} = 0.5V(i,j)(1 - V(i,j)) & \text{if} & \Delta V(i,j) < 0. \end{array} \tag{21}$$

Next states in Θ_{u2} are computed similarly to those of Θ_{v2}, the next-state functions being

$$u'((i,j)_{u2}) = \begin{cases} 0, & \text{if } \Delta U(i,j) < 0 \quad \& \quad (rand) < p_{u\to 0} \\ 1, & \text{if } \Delta U(i,j) > 0 \quad \& \quad (rand) < p_{u\to 1} \end{cases} \tag{22}$$

where

$$\Delta U(i,j) = U(i,j) - V(i,j)$$

probability values $p_{u\to 0}$ and $p_{u\to 1}$ in (22) are based on the following considerations:

- If $\Delta U(i,j) > 0$, prey is freely eaten. So, its density decreases with probability proportional to predatory density.
- If $\Delta U(i,j) < 0$, prey propagates with probability proportional to the number of the remainders

$$\begin{array}{lll} p_{u\to 0} = \Delta U(i,j) & \text{if} & \Delta U(i,j) > 0 \\ p_{u\to 1} = 0.5(\Delta U(i,j)(1 - \Delta U(i,j)) & \text{if} & \Delta U(i,j) > 0. \end{array} \tag{23}$$

Let $X = \{(i,j) : i = 0,\ldots,399, /j = 0,\ldots,799\}$. In the initial state, prey is spread over $\Omega_{u1}(0)$ with density, $U((i,j)_{u1}(0) = 0.4$ for all $(i,j)_{u1}) \in X_{u1}$. Predator has the density $V((i,j)_{v1}) = 0.1$ for the whole $\Omega_{v1}(0)$ except a band $\{(i,j)_{v1} : i = 0,\ldots,399, j = 369,\ldots,439\}$ where $V((i,j)_{v1}) = 1$ (Fig.5, t=0). Each tth iteration of the CA system consists of four following parts:

- V-Diffusion:
  1. $\Omega_{v1}(t)$ is copied to $\Omega'_{v1}(t)$.
  2. D_v iterations of $\aleph_{v1}$ are performed by application of Θ_{v1} to transfer from $\Omega_{v1}(t)$ to $\Omega_{v1}(t+1)$.
- U-Diffusion:
  1. $\Omega_{u1}(t)$ is copied to $\Omega'_{u1}(t)$.
  2. D_u iterations of $\aleph_{u1}$ are performed by application of Θ_{v1} to transfer $\Omega_{u1}(t)$ to $\Omega_{u1}(t+1)$.
- V-reaction:
  An iteration of $\aleph_{v2}$ is performed by applying Θ_{v2} (18) to all $(i,j)_{v2} \in X_{v2}$, probabilities being computed by (21), the resulting Ω_{v2} is copied to Ω_{v1}.

- U-reaction:
  An iteration of $\aleph_{u2}$ is performed by applying Θ_{u2} (22) to all $(i,j)_{u2} \in X_{v2}$, probabilities being computed from (23), the resulting Ω_{u2} is copied to Ω_{u1}.

In Fig. 5 four snapshots of the evolution of the predator component are shown, obtained by sequential implementation of the above four parts of the whole algorithm. CA system comes to its stable state rather slowly: the snapshot (t=500) is not yet close to it, but just that iteration number has been used for comparing times of sequential and parallel implementation.

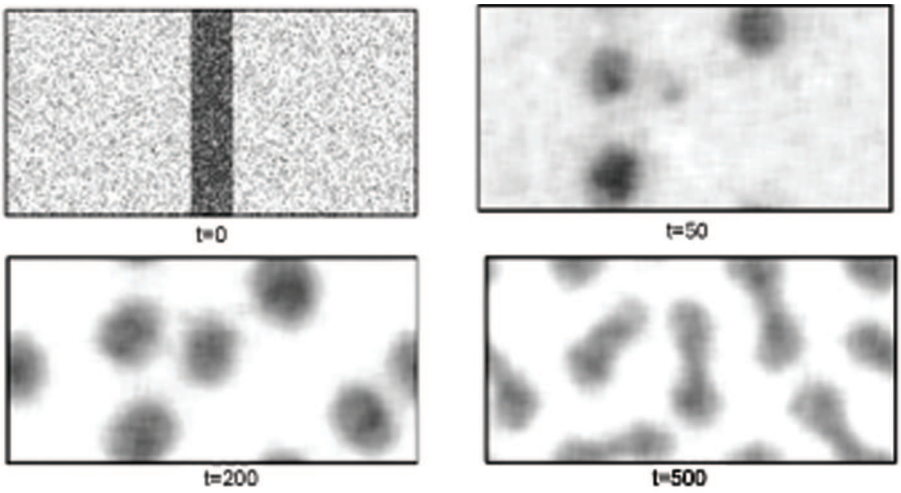

Fig. 5. Four snapshots of the evolution of predatory CA $\aleph_{v2}$ in prey-predatory CA system : initial cellular array $\Omega(0)$ in Boolean form, $\Omega(50)$, $\Omega(200)$ and $\Omega(500)$ – in averaged form

Implementation of the above CA system in two threads has been made by combining operation of $\aleph_{v1}$ and $\aleph_{v2}$ in one thread, and $\aleph_{u1}$ and $\aleph_{u2}$ — in the other thread. Implementation of the system in four threads has been performed by allocating each part of the above algorithm onto a thread. Due to the difference of the diffusion coefficients, the computation load in the threads differs essentially, which does not allow to obtain the perfect speedup. Implementation of a CA-system with equal thread load ($Dv = Du = 50$) results in the speedup is close to the number of threads. The obtained results with $T = 500$ iterations are compared to those obtained by the domain decomposition method, when each thread processes its own parts of two interacting cellular arrays, the thread loads being identical. The results of the above experiments are summarized in Table 1. It is seen from the table, that the speedup does not depend on the parallelization method, only the imbalance of thread load is essential.

Table 1. Computation time of 500 iterations and speedup of 1, 2, and 4 thread implementation of the CA -system of Example 2

number of threads	1		2		4	
	time	speedup	time	speedup	time	speedup
CA, Dv/Du=50	134	1	77	1.74	47	2.35
CA, Dv/Du=1	150	1	77	1.94	57	2.63
Dom-decomp	134	1	72	1.85	39	3.4

6 Conclusion

A concept of CA system is introduced, which is a set of CA working in common sharing common variables. Parallelization method for implementing CA-systems in multicore computer is presented. The method is based on associating each CA of the system to a thread of the parallel algorithm. Computational experiments were performed by simulating evolution of reaction-diffusion systems on a multicore computer of the type Intel i7. They show, that the speedup depends on the imbalance of the load between threads. It is equal to n (the number of cores used) when computational load in threads is identical, and quite acceptable ($> 0.8n$) when the load ratio is $0.8 < \rho < 1.2$. Comparison of obtained efficiency with that of parallelization using domain decomposition resulted in minor difference in the above interval of load imbalance.

References

1. Wolfram, S.: Cellular Automata abd Comlexity – Collected paopers. Addison Wesley, Reading (1994)
2. Hoekstra, A.G., Kroc, J., Sloot, P.M.A.,(eds.):: Simulating Complex Systems by Cellular Automata. Understanding complex Systems. Springer-Verlag, Berlin (2010)
3. Bandman, O.: Cellular Automarta Composition Techniques for Spatial Automata Simulation. In: Hoekstra, A.G., Kroc, J.,Sloot, P.M.A.,(eds.) Simulating Complex Systems by Cellular Automata. Understanding complex Systems. : Springer-Verlag, Berlin, (2010) 81-115
4. Achasova, S., Bandman, O., Markova, V., Piskunov, S.: Parallel Substitution Algorithm. Theory and Application. World Scientific, Singapore(1994)
5. Bandman,O.: Coarse-Grained Parallelization of Cellular-Automata Simulation Algorithm. In: Malyshkin,V. (ed.) PaCT 2007 Proc., LNCS, vol. 4671, Springer, Berlin (2007) 370-384
6. Toffolli, T.,Margolus, N.: Cellular Automata Machine. MIT Press, USA (1987)
7. Deutsch, A., Dorman, S.: Cellular Automata Modeling of Biological Pattern Formation. Birkhäuser, Berlin (2005)
8. Cataneo, G., Dennunzio, A., Farina, F. :A Full Cellular Automaton to Simulate Predatory-Prey Systems. LNSC, 4273 (2006) pp.446-451

9. Chua, L.: CNN: a paradigm of complexity. World Scientific,Singapore (2002).
10. Bandman, O.: Simulating Spatial Dynamics by Pribabilistic Cellular Automata. In: Bandini, S., B.Chopard, B., Tomassini, M.(eds.) ACRI-2002, LNCS, vol.2493. Springer, Berlin ,(2002)10-16