Bull. Nov. Comp. Center, Comp. Science, 10 (1999), 1–14
© 1999 NCC Publisher

Application of Parallel Substitution Algorithm for spatial dynamics simulation

O. Bandman

Parallel Substitution Algorithm (PSA) is a formal model of fine-grained parallel computation. It has been developed and used for design and investigation of highly parallel algorithms and digital systems architecture. Here it is shown, that its expressive capabilities allow also to use PSA formalisms for representing a wide range of spatially distributed algorithms. As illustrations, PSA of discrete and continuous, synchronous and asynchronous, deterministic and stochastic models of spatial dynamics are presented.

1. Main features of PSA

PSA has evolved when algorithmically oriented highly parallel architecture design methods came into demand. The attempt to use Cellular Automaton for this purpose showed that it lacks of means for formal representation of complex digital devices and for creation computer aided design tools for elaborating and simulating fine-grained cellular algorithms. So, CA was strongly enriched by including in it some additional expression capabilities. The new model was called Parallel Substitution Algorithm, or PSA for short [1]. After some experience of PSA exploiting for discrete algorithms investigation it became clear that PSA may be successfully used also for modeling a wide class of natural phenomena used to be described by Partial Differential Equations (PDE) or Artificial Neural Networks. The following properties of PSA make it powerful.

• PSA processes *cellular arrays* which are sets of cells $W \in A \times M$, where A is a finite alphabet, M is a *naming set* (in general case a countable one). A cell (a, m) is characterized by a state $a \in A$ and a name $m \in M$. The state is an abstraction of a data item, it may be a symbol, a number, a vector, an image. The name is a label assigned to a processing element. The most frequently used naming set is the set of Cartesian coordinates. On the set M a function $\phi : M \to M$ is defined, $\phi(m)$ being a neighbour for any $m \in M$. A set of naming functions form a *template* $T(m) = \{\phi_j(m) : j = 0, \ldots, q\}$ which for a given cell $(a_0, m_0) \in W$ determines its *neighborhood*

$$N(m_0) = \{(a_0, m_0), \ldots, (a_1, \phi_j(m_0)) : \ j = 1, \ldots, q\}. \tag{1}$$

• Operations over a cellular array are specified by a set $\Phi = \{\Theta_i\}$, $i = 1, \ldots, n$, of parallel substitutions

$$\Theta_i : C_i(m) * S_i(m) \to S_i'(m). \tag{2}$$

In (2), $C_i(m)$, $S_i(m)$ and $S_i'(m)$ are *local configurations*

$$\begin{aligned} C_i(m) &= \{(y_{ik}, \phi_{ik}(m)) : k = 0, \ldots, q_y\}, \\ S_i(m) &= \{(x_{ij}, \phi_{ij}(m)) : j = 0, \ldots, q_x\}, \\ S_i'(m) &= \{(f_{il}(X,Y), \phi_{il}(m) : l = 0, \ldots, q_f\}, \end{aligned} \tag{3}$$

where

1) no pair of naming functions (ϕ_{ij}, ϕ_{ik}) are identical,
2) $x_{ij} \in X$, $y_{ik} \in Y$ are state variables or constants and $f_{il}(X,Y)$ are cellular functions with the domain from A.

• A substitution *is applicable* to $W \in A \times M$, if there is at least one cell named $m_0 \in M$ such that

$$C_i(m_0) \cup S_i(m_0) \subseteq W, \tag{4}$$

cells (x, m) being thought of as included in W, if $(a, m) \in W$ and $a \in A$. Application of a substitution at a cell $(a, m_0) \in W$ (m_0-*microoperation*) is the replacement of the subset $S_i(m_0) \in W$ by $S_i'(m_0)$, i.e.,

$$W_{m_0}(t+1) = (W \setminus S_i(m_0)) \cup S_i'(m_0). \tag{5}$$

A local configuration $S_i(m)$ in a substitution is called its *base*. It determines the neighbors of a cell named m which are changed, when the substitution is applied. A local configuration $C_i(m)$ is not changed during the substitution. So, it is called *a context.*

• There are three modes of parallel substitutions application.

1) *Synchronous mode*, when all substitutions are applied at all cells at each step at once. At this case in order to provide determinism of the computation, one should be careful not to allow the substitutions be contradictory when $|S_i'(m)| > 1$ [1].

2) *Asynchronous mode*, when any substitution is applied at a cell when it is applicable, only one application being allowed at a time. This mode is used when only an averaged result is needed to be determinate.

3) *n-steps synchronous mode*, when each time-step is divided into n substeps executed in turn, all substitutions of each subset being applied to a subset of cells at each substep at once.

• A Parallel Substitution Algorithm is defined by a set of substitutions together with the indication of the mode of application. Implementation of a PSA over a cellular array W is the iterative procedure, where at each step a set of microoperations (5) is executed at a set of cells, according to the given mode. The algorithm stops at a step T if no substitutions are applicable to the cellular array $W(T)$, i.e., $W(T+1) = W(T)$.

• If in the substitution set there is at least one Θ_i, such that $|S_i(m)| < |S_i'(m)|$, then the cardinality of a cellular array under processing is growing during the PSA execution. If, in addition to that, $T_i(m) \subset T_i'(m)$, then the growth of the array is without the loss of a single cell name, $T_i(m)$ and $T_i'(m)$ being the underlying templates of $S_i(m)$ and $S_i'(m)$, respectively. Similarly, when $T_i'(m) \subset T_i(m)$ the array diminishes in size and may completely disappear. Such type of substitutions determines the *nonstationary* PSAs which are used in modeling natural processes, especially the artificial life. If all substitutions $\Theta_i \in \Phi$ meet the condition

$$T_i(m) = T_i'(m), \tag{6}$$

they are called *stationary*. They are used to model processes in cellular array of fixed structure.

• A PSA may process not only one but a few arrays $\boldsymbol{W} = \{W_1, \ldots, W_l\}$ as well. In this case, each substitution Θ_i is allowed to be applied to only one array. It means that its base $S_i(m)$ is located in only one $W_j \in \boldsymbol{W}$, i.e., $m \in M_j$. As for the context $C_i(m)$, it may be located at any array, moreover, it may be composed of a number of local configurations, located in different arrays, i.e.,

$$C_i(m) = C_{i1}(m_{j1}) * \ldots * C_{ik}(m_{jk}), \quad k \leq l. \tag{7}$$

2. PSA representation of diffusion models

2.1. PSA of finite-difference representation of diffusion PDE

Diffusion is a process which aims at a stable distribution of concentration in a system which is the result of a disordered wandering of system elements. In the simple two-dimensional case diffusion is represented by the following Partial Differential Equation (PDE)

$$\frac{\partial u}{\partial t} = d\Delta u, \tag{8}$$

where $\Delta u = \frac{\partial^2 u}{\partial^2 x} + \frac{\partial^2 u}{\partial^2 y}$ is a Laplacian, $u(t, x, y)$ is a concentration, d is a diffusion coefficient (assumed to be constant), x, y are the Cartesian coordinates of the plane space. When solving this equation by finite-difference method, time and space are discretized, so that $x = ih_x$, $y = jh_y$, and $t = n\tau$, i, j, n being integers. If $h = h_x = h_y = 1$, equation (1) takes the following discrete form:

$$u_{i,j}(t+1) = u_{i,j} + \tau d(u_{i-1,j} + u_{i+1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}). \qquad (9)$$

In (9) and further, $u(t)$ is written as u for short. Since time and space in finite difference methods are discrete, the PSA representation is admissible. It is obtained as follows:

1. The alphabet is the set of all real numbers $A = \boldsymbol{R}$, variables u_k ($k = 0, \dots, 4$), and functions $u_k(t+1)$ having the same domain.

2. The naming set is a set of coordinate pairs of a 2D Cartesian lattice of finite size $P \times Q$, $M = \{\langle i,j \rangle : \ i = 0, \dots, P, \ j = 0, \dots, Q\}$.

3. The substitution set consists of a single stationary substitution

$$\begin{aligned}\Theta : \{&(u_1, \langle i+1, j-1 \rangle)(u_2, \langle i, j+1 \rangle)(u_3, \langle i+1, j \rangle)(u_4, \langle i-1, j \rangle)\} * \\ &(u_0, \langle i,j \rangle) \to (u_0(t+1), \langle i,j \rangle), \qquad (10)\end{aligned}$$

with

$$u_0(t+1) = u_0 + D(u_1 + u_2 + u_3 + u_4 - 4u_0),$$

where D is the normalized diffusion coefficient, $D = d\tau$. The choice of τ depends on convergence conditions which are known to be met with $(d\tau)/(h^2) < 1/4$. Hence, further it is reasonable to take $D = 1/4$ with $h = 1$.

2.2. PSA of asynchronous deterministic cellular automata diffusion model

There are several CA models of diffusion [2]. To ensure that they are adequate to real life diffusion, the transition rules should provide:

1) the conservation of mass and momentum, i.e., the number of "ones" in the array should be constant during the computation process,

2) the symmetry in space, i.e., the rule should be invariant relative to a rotation at a certain angle.

One of the most primitive model, satisfying the above conditions is a "naive diffusion", proposed in 1D variant in [3]. A 2D variant has been described and investigated in [2].

Informally CA model is as follows: each cell interchanges states with one of its neighbor with the probability $p = 1/q$, where q is the amount of its neighbors. Such a rule being applied to an array in synchronous mode is "contradictory", i.e., two different state values may occur in a single cell, resulting in disappearance or in emergency of "particles" which is in contradiction of the conservation laws. So, only asynchronous mode of computation is admissible. To provide a random choice (with $p = 1/4$) of a neighbour with whom to exchange states, and to provide a random choice of the cell in the array where the substitution is applied, two special context cells should be used. Hence, PSA acts on a composition of 3 cellular arrays:

1. The main array with cells named from $M_0 = \{\langle i,j\rangle : i = 0,\ldots,P-1;\ j = 0,\ldots,O-1\}$, and the alphabet $A_0 = \{0,1\}$ with a state variable u, representing the concentration of the substance under the diffusion.
2. A single cell context array for the random choice of the cell at which a substitution is applied. The naming set $M_1 = \{m_0\}$, the alphabet is equal to the naming set of the main array $A_1 = \{\langle i,j\rangle : i = 0,\ldots P-1;\ j = 0,\ldots,O-1\}$. The cell $((i,j),m_0)$ has two inputs connected to the generators of random numbers.
3. A single cell context array for random choice of the neighbour with the naming set $M_2 = \{m_1\}$ and the alphabet $A_2 = \{N,E,S,W\}$, state variable being denoted as X. The cell (X,m_1) has an input from a generator of random numbers.

The substitution set is represented by a single parallel substitution

$$\begin{aligned}\Theta_X &= \underbrace{((i,j),m_0)}_{C_1(t)} * \underbrace{(X,m_1)}_{C_2(t)} * \underbrace{\{(u_0,\langle i,j\rangle)(u_X,\phi_N(i,j))\}}_{S_X(t)} \\ &\rightarrow \underbrace{\{(u_X,\langle i,j\rangle)(u_0,\phi_N(i,j))\}}_{S_X(t+1)}. \end{aligned} \tag{11}$$

Graphical representation of Θ_N, Θ_E, Θ_S, Θ_W is as follows:

$$\Theta_N : \boxed{i,j} * \boxed{N} * \begin{array}{|c|}\hline x_1 \\ \hline x_0 \\ \hline\end{array} \rightarrow \begin{array}{|c|}\hline x_0 \\ \hline x_1 \\ \hline\end{array};\ \Theta_W : \boxed{i,j} * \boxed{E} * \begin{array}{|c|c|}\hline x_2 & x_0 \\ \hline\end{array} \rightarrow \begin{array}{|c|c|}\hline x_0 & x_2 \\ \hline\end{array};$$

$$\Theta_S : \boxed{i,j} * \boxed{S} * \begin{array}{|c|}\hline x_0 \\ \hline x_3 \\ \hline\end{array} \rightarrow \begin{array}{|c|}\hline x_3 \\ \hline x_0 \\ \hline\end{array};\ \Theta_E : \boxed{i,j} * \boxed{W} * \begin{array}{|c|c|}\hline x_0 & x_4 \\ \hline\end{array} \rightarrow \begin{array}{|c|c|}\hline x_4 & x_0 \\ \hline\end{array}.$$

In order to obtain the real (physical) values U_{ij} of concentration, the procedure of *averaging* is performed. It consists of taking the average of total amount of the Boolean "ones" on a square $r \times r$ around each cell:

$$U_{ij} = \frac{1}{r^2}\sum_{i=1}^{r}\sum_{j=1}^{r} u_{ij}. \tag{12}$$

PSA simulation of naive diffusion has been used in order to find the diffusion coefficient D_{naive} of the model. The diffusion coefficient has been determined by comparing the function of averaged concentration distribution $U_{\text{naive}}(j)$ with the similar one $U_{\text{PDE}}(j)$, calculated according to (10). Both functions are taken along the j-axis going via the center of the array ($i = 0, j = 0$). A pair of time numbers ($T_{\text{naive}}, T_{\text{PDE}}$), such that two functions: $U_{\text{naive}}(j)$ at T_{naive} and $U_{\text{PDE}}(j)$ at T_{PDE} coincide, are referred to as *compatible times*. The diffusion coefficient D_{naive} is computed according to the following formula with $D_{\text{PDE}} = 0.25$.

$$D_{\text{naive}} = D_{\text{PDE}}\frac{T_{\text{naive}}}{T_{\text{PDE}}} = 0.25\frac{T_{\text{naive}}}{T_{\text{PDE}}}, \tag{13}$$

where T_{naive} and T_{PDE} form a pair of compatible times.

The simulation experiments showed that D_{naive} is not constant, it decreases with the decrease of concentration according to the following law:

$$D_{\text{naive}}(u) = 0.5(1 - e^{-u}).$$

2.3. PSA of stochastic synchronous cellular automata diffusion model

A stochastic CA diffusion model proposed in [3] and referred to as Block-Rotation model in [2] is meant to be executed in two-step synchronous mode. Accordingly, a partition of the array into two parts is to be constructed, both parts consisting of blocks 2 × 2 cells (Figure 1).

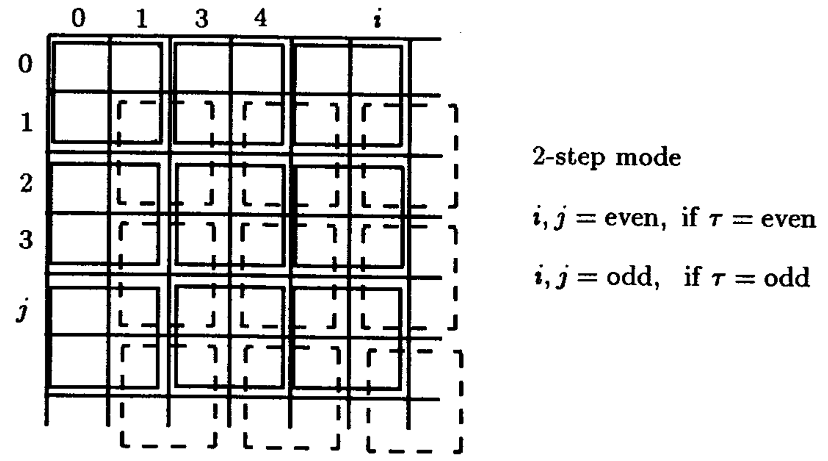
0
1
3
4
i
0
1
2
3
j
2-step mode
i, j = even, if τ = even
i, j = odd, if τ = odd

Figure 1. Partitioning of a cellular array into even and odd parts

The first part is called the *even part*, its blocks have even coordinates of their top right cells. The second is referred to as the *odd part* its blocks have top right cells with odd coordinates. Such type of CA partitioning is called a *Margolus neighborhood* [3]. Each iteration is divided into two steps. At the even steps the transition rule is applied to all even blocks, at the odd steps –

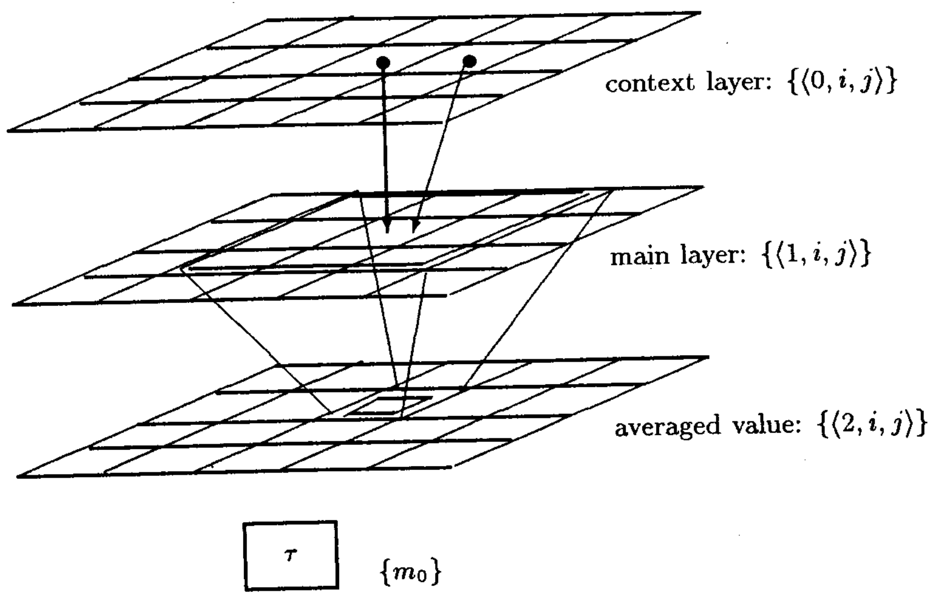
context layer: $\{\langle 0,i,j\rangle\}$
main layer: $\{\langle 1,i,j\rangle\}$
averaged value: $\{\langle 2,i,j\rangle\}$
τ
$\{m_0\}$

Figure 2. Architecture of Block-Rotation diffusion PSA

to all odd blocks. Such an alternating of even and odd parts in execution process allows to avoid contradictoriness. The transition rule is one and the same at odd and even steps: at each even (odd) step all even (odd) blocks are rotated $\pi/2$ clockwise or counterclockwise with equal probability p.

To investigate the model a composition of two cellular array is chosen: 1) a three layer array $M = \{\langle k,i,j\rangle\}$, $k = 0,1,2$, $i,j = 0,\ldots,Q-1$ and 2) a single cell array $\{m_0\}$ (Figure 2).

According to PSA architecture the parallel substitution set contains the following substitution groups.

The substitution representing generation of a timing series which controls the alternation of even and odd parts of the PSA.

$$\Theta_\tau = (\tau, m_0) \to (\bar{\tau}, m_0). \tag{14}$$

The substitution acting on the 0-th context array which performs the partitioning of the cell set into two parts:

$$\begin{aligned}\Theta_0 : \ &\{((i,j),\langle 0,i,j\rangle)(\text{rand},\langle 0,i,j+1\rangle)\}\\ &\to \{(f_1,\langle 0,i,j\rangle)(f_2,\langle 0,i,j+1\rangle)\},\end{aligned} \tag{15}$$

where

$$f_1 = \begin{cases} \text{ev, if} & i = \text{even} \ \& \ j = \text{even},\\ \text{od, if} & i = \text{odd} \ \& \ j = \text{odd},\end{cases} \qquad f_2 = \begin{cases} \text{cl,} & \text{if rand} < 1/2,\\ \text{co,} & \text{if rand} \geq 1/2.\end{cases} \tag{16}$$

The main diffusion process is represented by the set $\{\Theta_1,\Theta_2,\Theta_3,\Theta_4\}$, acting on the first layer of M.

$$\begin{aligned}
\Theta_1 : \; & (1, m_0) * (\mathrm{ev}, \langle 0, i, j\rangle) * (\mathrm{cl}, \langle 0, i, j+1\rangle) * \\
& \{(u_1, \langle 1, i, j\rangle)(u_2\langle 1, i+1, j\rangle)(u_3, \langle 1, i+1, j+1\rangle)(u_4, \langle 1, i+1.j\rangle)\} \\
& \rightarrow \{(u_1, \langle 1, i+1, j\rangle)(u_2\langle 1, i+1, j+1\rangle) \\
& \quad (u_3, \langle 1, i+1, j\rangle)(u_4, \langle 1, i, j\rangle)\}; \qquad (17) \\
\Theta_2 : \; & (1, m_0) * (\mathrm{ev}, \langle 0, i, j\rangle) * (\mathrm{co}, \langle 0, i, j+1\rangle) * \\
& \{(u_1, \langle 1, i, j\rangle)(u_2\langle 1, i+1, j\rangle)(u_3, \langle 1, i+1, j+1\rangle)(u_4, \langle 1, i+1.j\rangle)\} \\
& \rightarrow \{(u_1, \langle 1, i, j+1\rangle)(u_2\langle 1, i+1, j+1\rangle) \\
& \quad (u_3, \langle 1, i+1, j\rangle)(u_4, \langle 1, i, j-1\rangle)\}.
\end{aligned}$$

In the graphical form Θ_1 and Θ_2 are as follows:

$$\Theta_1 : \; \boxed{1} * \begin{array}{|c|c|}\hline \mathrm{ev} & \mathrm{cl} \\ \hline\end{array} * \begin{array}{|c|c|}\hline u_0 & u_1 \\ \hline u_3 & u_2 \\ \hline\end{array} \;\Rightarrow\; \begin{array}{|c|c|}\hline u_3 & u_0 \\ \hline u_2 & u_1 \\ \hline\end{array}$$

$$\Theta_2 : \; \boxed{1} * \begin{array}{|c|c|}\hline \mathrm{ev} & \mathrm{co} \\ \hline\end{array} * \begin{array}{|c|c|}\hline u_0 & u_1 \\ \hline u_3 & u_2 \\ \hline\end{array} \;\Rightarrow\; \begin{array}{|c|c|}\hline u_1 & u_2 \\ \hline u_0 & u_3 \\ \hline\end{array}$$

The substitutions Θ_3 and Θ_4 differ from Θ_3 and Θ_4 in states of cells named $\langle 0, i, j\rangle$ $\langle 0, i, j+1\rangle$ which are 0 and **od** instead of 1 and **ev**, respectively.

The substitutions for computing the averaged values at even steps.

$$\begin{aligned}
\Theta_2 \; &= \; (1, m_0) * \{(u_{kl}, \langle 2, i+k, j+l\rangle) : k, l = -r/2, \ldots, r/2\} \\
& \rightarrow (F(u_{kl}, \langle 2, i, j\rangle), \qquad (18)
\end{aligned}$$

where $F(u_{kl})$ is calculated according to (12).

The simulation of Block Rotation model has been performed to determine the dependence of the diffusion coefficient $D = \tau d$ from the concentration value. In [4] D is proved theoretically to be equal to $D_R = 3/2$ for the probability $p = 1/2$. This value characterizes the abstract diffusion model, but is too large to be used for simulating any physical process. So, it is necessary to know how to modify the transition rules to obtain a model of a real required process. Moreover, for the comparison of simulation results with PDE solutions considered to be correct, the diffusion coefficients should be equal to that of used in PDE, and, hence, it should meet condition (13).

There are three possibilities to regulate the diffusion coefficient $D = d\tau/h^2$ of the model:

1. Variation of time step τ. For example, if a coefficient D is wanted, then the time step is $\tau' = D_R\tau/D = 3/2D$. So, if $D = 0.25$, τ' is to be 6 times less than τ, i.e., each step should be considered as $\tau/6$.
2. Variation of the probability p. Taking $p' = pD_R/D$, it is possible to model process with D as a diffusion coefficient.
3. Variation of spatial step h which should be taken as $h' = h\sqrt{D_r/D}$.

3. PSA of Cellular Neural Networks

3.1. PSA representation of Cellular Neural Associative Memory

Cellular Neural Associative Memory (CNAM) [5] is a variant of the well-known Hopfield's neural network paradigm [6], differing from the latter in connection structure which is local for CNAM like that of cellular automata. Storage and retrieval capabilities of CNAM have been investigated, as well as a number of learning methods have been proposed in several papers [7].

CNAM is a rectangular array of neurons which are bistable elements with states from $\{-1, 1\}$. Each neuron interacts with its neighbours through weighted connections, weights values being real numbers. The neighbourhood structure is one and the same for all neurons, but connection weight values are different. The operation mode of CNAM is iterative and synchronous. At each step all neurons calculate their next states which are the values of a nonlinear (threshold) function of the weighted sum of their neighbours states. The operation stops when a stable state of CNAM is reached, i.e., no state changes occur any more. Operating in such a way CNAM performs the retrieval of one of the patterns, stored in it in the form of global stable states (attractors) and called *prototypes*. Correspondence between a given set of prototypes and the set of attractors is provided by the connection weights values which are determined by means of a learning procedure [7]. When any pattern is input by setting the CNAM in an initial state, the operation starts. It terminates in the stable state which is equal to the prototype with the closest resemblance to the input pattern.

The PSA representation of CNAM is as follows:

1. The alphabet is combined from two sets: $A = A_p \cup A_w$, where $A_p = \{-1, 1\}$ serves for pattern (neuron states) representation, $A_w = \boldsymbol{R}$ is used to represent weight values.

2. The naming set is a multilayer array $M = \{\langle k, i, j\rangle\}$, $k = 0, \ldots, q$ representing layer numbers, $i = 0, \ldots, P-1$ and $j = 0, \ldots, Q-1$ are the numbers of rows and columns, respectively. The 0-th layer M_0 contains the cells n_{ij} called *neurons*, their states are from A_p. The layers from $M_w = \boldsymbol{M} \backslash M_0$ numbered as $k = 1, \ldots, q$ have cells whose states are from $\boldsymbol{R}$, so that

the cell state named $\langle k,i,j\rangle$ is equal to the weight $w_k(i,j)$ of the connection between the neuron n_{ij} and its k-th neighbour.

3. The following naming functions determine the connection structure of the array. Each neuron n_{ij} is connected with the cells from its neighbourhood in M_0, defined by the template

$$T_0(i,j) = \{\phi_1(0,i,j),\ldots,\phi_q(0,i,j)\}. \tag{19}$$

To use the connection weights for calculation its next state the neuron n_{ij} should also be connected to the cells defined by the template

$$T_w(k) = \{\langle 1,i.j\rangle \ldots,\langle q,i.j\rangle\}. \tag{20}$$

4. The computation process is represented by the following substitution

$$\begin{aligned}\Theta_0 : C_0(\langle k,i,j\rangle) * C_w(\langle k,i,j\rangle) * \{(x_0,\langle 0,i,j\rangle)\}\\ \rightarrow \{(F(X(i,j),W(i,j)),\langle 0,i,j\rangle)\},\end{aligned} \tag{21}$$

where

$$\begin{aligned}C_0(\langle 0,i,j\rangle) &= \{(x_1(i,j),\phi_1(0,i,j)),\ldots,(x_q(i,j),\phi_q(0,i,j))\},\\ C_w(\langle k,i,j\rangle) &= \{(w_k(ij),\langle k,i.j\rangle\},\ \ k=1,\ldots,q,\\ F(X(i,j),W(i,j)) &= \begin{cases}1 & \text{if } \sum_{k=1}^{q} x_k(ij)w_k(ij) \geq 0,\\ -1 & \text{otherwise.}\end{cases}\end{aligned}$$

In (21), the first two sets are contexts generated by the templates T_0 and T_w respectively (Figure 3), $X(i,j) = \{x_1(i,j),\ldots,x_q(i,j)\}$ is the set of neighbourhood sell states, $W(i,j) = \{w_1(i,j),\ldots,w_q(i,j)\}$. The base is a single cell local configuration with the naming space domain $M_0' = \{\langle 0,i.j\rangle\}$, $i = 1,\ldots,P-1$, $j=1,\ldots,Q-1\}$.

If the automatic stop of the algorithm is wanted, then an additional layer (let it be named $k=-1$), a counter named c, and a signal cell named s should be added. States from the 0-th layer are rewritten every v iteration in (-1)-th layer, v being the counter state. When two successive patterns on this layers coincide, the signal cell transits in the state "stop". The automatic stop algorithm is represented by the following substitutions:

$$\begin{aligned}&\Theta_1 : \ \{(v,c)\} * \{x_{ij},\langle 0,i,j\rangle\} * \{y_{ij},\langle -1,i,j\rangle\} \rightarrow \{x_{ij},\langle -1,i,j\rangle\},\\ &\Theta_2 : \ \{(v,c+1)\} * \{x_{ij},\langle 0,i,j\rangle\} * \{y_{ij},\langle -1,i,j\rangle \rightarrow \{(f(x,y),s)\},\\ &\Theta_3 : \ \{(v,c)\} \rightarrow \{((v+1)_{\text{mod}\,v},c)\},\end{aligned} \tag{22}$$

where

$$f(x,y) = \begin{cases}\text{"stop"} & \text{if } x_{ij} = y_{ij} \text{ for all } \langle 0,i,j\rangle \in M',\\ 0 & \text{otherwise}\end{cases}$$

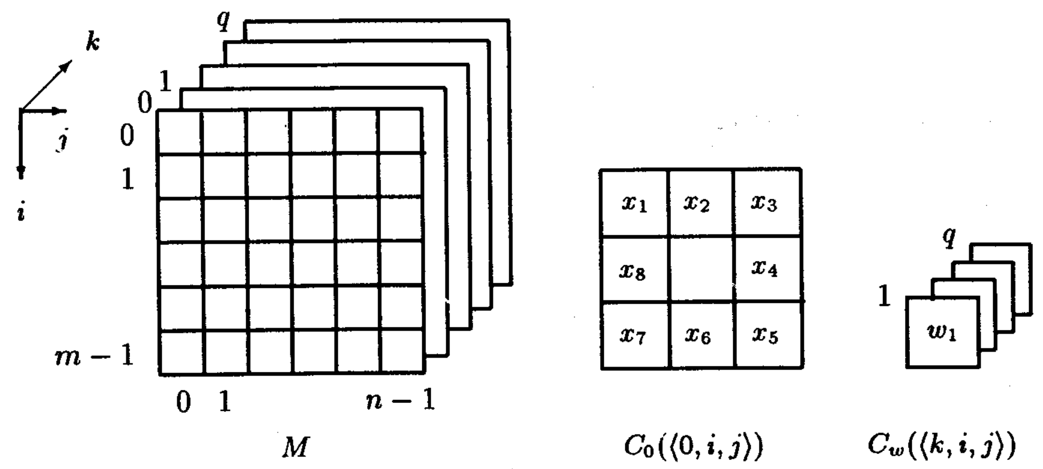

Figure 3. Graphical representation of the naming set and the local configuration of the substitution (21)

The learning process of CNAM by the method, proposed in [5], may be represented by the PSA with the same architecture as the CNAM to be taught. This fact significantly simplifies CNAM application both for software development and all the more for designing the special purpose hardware.

In the learning process, the cells storing connection weights are iteratively updated according to the given prototypes which are sequentially input into the neuron layer. Hence, in the main substitution of the learning PSA the "neuron" (M_0) and the "weight" (M_w) subarrays interchange their roles. A local configuration generated by T_w in the "weight" layers plays the role of the base, while local configuration, generated by T_0 in the "neuron" layer is a context one (the process of prototypes input into M_0 is not considered). So, the learning substitution is as follows:

$$\Theta_w : C_0(\langle 0,i,j\rangle) * S_w(\langle k,i,j\rangle) \to S'_w(\langle k,i,j\rangle), \tag{23}$$

where $C_0(\langle 0,i,j\rangle)$ is identical to (22),

$$\begin{aligned} S_w(\langle k,i,j\rangle) &= \{(w_k(i,j),\langle k,i.j\rangle\}, \quad k=1,\dots,q,\\ S'_w(\langle k,i,j\rangle) &= \{(f,\langle k,i.j\rangle\}, \quad k=1,\dots,q, \end{aligned}$$

and

$$f = \begin{cases} w_k(i,j) & \text{if } x_0(i,j)\sum_{k=1}^{q} x_k(i,j)w_k(i,j) > 0,\\ w_k(i,j) + x_0(i,j)x_k(i,j) & \text{otherwise.} \end{cases}$$

3.2. PSA representation of two-layer Cellular Neural Network

Cellular Neural Networks (CNN) [8] are fine-grained parallel systems having all properties of Artificial Neural Networks but the connection structure which is local like in Cellular Automata. Like in Cellular Automata a CNN cell is characterized by the internal state $x \in \boldsymbol{R}$, an output state $y \in [-1, 1]$, and a set of inputs which are output states of neighbouring cells. Each cell calculates the next state $x(t+1)$ as a weighted sum of the output states of neighbouring cells, and a next output as a sigmoid-like nonlinear function $y(t+1) = f(x(t))$ which is taken as a piece-wise approximation of the form

$$f(x) = \frac{|x+1| + |x-1|}{2}. \tag{24}$$

CNN may be regarded as a finite difference representation of reaction-diffusion Partial Differential Equations: a single layered CNN modeling the first order phenomena, two-layer CNN – second order ones. The latter exhibit autowaves propagation [9]. Connection weight allocation to the neighbours is identical for all cells in each layer, weight values determining the type of generated autowave (travelling front, travelling impulse, chaotic behavior).

Second order CNN is represented usually by a system of two finite-difference equations as follows:

$$\begin{aligned} x^1(t+1) &= \alpha x^1(t) + ay^2(t) + \sum_{l=1}^{q_1} w_l^1 y_l^1(t)), & (25)\\ x^2(t+1) &= \beta x^2(t) + by^1(t) + \sum_{l=1}^{q_2} w_l^2 y_l^2(t)), & (26)\\ y^k &= F(x^k), \quad k = 1, 2. & (27) \end{aligned}$$

PSA representation is as follows:

1. The alphabet consists of two subsets $A_0 \cup A_w$. The domain of A_0 is equal to the interval $[-1, 1]$, that of A_w is $\boldsymbol{R}$ for representing connection weights and internal states.

2. The naming set is a four layer array: two layers, $k = 1_x$ and $k = 2_x$, for storing internal states, and two others, $k = 1_y$ and $k = 2_y$, for outputs: $M = \{\langle k, i, j\rangle\}$, $k = 1_x, 1_y, 2_x, 2_y$, $i = 0, \ldots, P-1$, $j = 0, \ldots, Q-1$.

3. Connection structure is given by 3 templates, $T_x^k(i,j)$, $T_y^k(i,j)$, $k = 1, 2$; determining cell interactions in the first and in the second layers, respectively:

$$\begin{aligned} T_x^k(i,j) = \{&\langle k_x, i, j\rangle, \langle k_y, i-1, j\rangle, \langle k_x, i, j+1\rangle,\\ &\langle 1_x, i+1, j\rangle, \langle 1_x, i, j-1\rangle, \langle (k+1)_{\mathrm{mod}\,2}, i, j\rangle\}, \end{aligned} \tag{28}$$

$$T_y^k(i,j) = \{\langle k_x, i, j\rangle\}, \quad k = 1, 2. \tag{29}$$

4. PSA is represented by the following substitutions operating in a two-step synchronous mode. At the even steps ($\tau = 0$) the internal states are calculated and at the odd steps ($\tau = 1$) the outputs are obtained according to (27)

$$\begin{aligned} \Theta_x^k : \quad & (0,\tau) * C_x^k(i,j) * C_y^k(i,j) * \{(x^k, (\langle k_x, i, j\rangle)\} \to \\ & \{(F_x^k(X,Y), \langle k_x, i, j\rangle)\}, \\ \Theta_y : \quad & (1,\tau) * \{(x^k, (\langle k_x, i, j\rangle)\} * \{(y^k, \langle y_y^k, i, j\rangle)\} \to \\ & \{(f(x^k), (\langle y_y^k, i, j\rangle)\}, \end{aligned} \tag{30}$$

where $C_x^k(i,j)$ and $C_y^k(i,j)$ are local configurations generated by the templates $T_x^k(i,j)$ and $T_y^k(i,j)$, respectively,

$$\begin{aligned} C_x^k(i,j) &= \{(x_0^k, \langle k_x, i, j\rangle), (y_1^k, \langle y_1, i-1, j\rangle), (x_2^k, \langle k_x, i, j+1\rangle), \\ &\quad (x_3^k, \langle k_x, i+1, j\rangle), (x_4^k, \langle k_x, i, j-1\rangle)\}, \\ C_y^k(i,j) &= \{y^{(k+1)_{\mathrm{mod}\,2}}, \langle k_x, i, j\rangle\}, \quad k = 1, 2. \end{aligned}$$

The structure of CNN connections is shown in Figure 4.

The function $F_x^k(X,Y)$ is as follows

$$F_x^k(X,Y) = \alpha x_0^k + \sum_{l=1}^{q_k} w_l y_l^k + a y^{(k+1)_{\mathrm{mod}\,2}}.$$

The weights α, a, β, b, $w_1^k, \ldots, w_2^k$, $k = 1, 2$, are as in (25), (26).

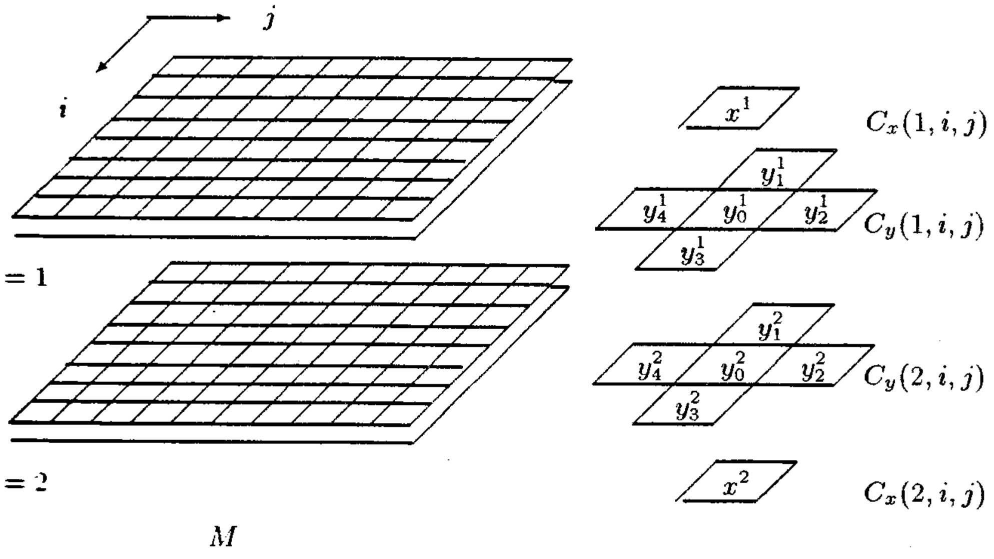

Figure 4. The naming set M and the local configurations of CNN PSA

4. Conclusion

It is shown that the formalisms of PSA is a convenient and universal tool for representing a wide range of spatially distributed processes. The unified form of parallel substitutions for representing binary and continuous, synchronous and asynchronous, elementary and functional processes allows to use one and the same simulation system for designing and investigation all kinds of spatial dynamics. A special purpose simulation tool WinALT [10] based on PSA concepts supports PSA use.

References

[1] Achasova S., Bandman O., Markova V., Piskunov S. Parallel Substitution Algorithm. Theory and Application. – Singapore: World scientific, 1994.

[2] Bandman O. Comparative study of Diffusion Cellular Automata Models // Lecture Notes in Computer Science. – 1999. – Vol. 1662. – P. 395–409.

[3] Tpffolli T., Margolus N. Cellular Automata Machine // MIT Press. – 1987.

[4] Malinetsli G.G., Stepantsov M.E. Modelling Diffusive Processes by Cellular Automata with Margolus Neighbourhood // Zhurnal Vychislitelnoj Matematiki i Matematicheskij Physiki. – 1998. – Vol. 38, No. 6. – P. 1017–1021 (in Russian).

[5] Bandman O.L., Pudov S.G. Stability of stored patterns on Cellular-Neural Associative memory // Bulletin of Novosibirsk Computing Center, issue Comp. Sci. – 1996. – № 4. – P. 3–20.

[6] Hopfield J.J., Tank D.W. Computing with Neural Circuits: a Model // Science. – 1986. – Vol. 233. – P. 625–638.

[7] Pudov S. Comparative Analysis of Learning Methods of Cellular-Neural Associative Memory // Lecture Notes in Computer Science. – 1999. – Vol. 1662. – P. 188–199.

[8] Chua L. Cellular Neural Networks: the Paradigm of Complexity. – Singapore: World Scientific, 1999.

[9] Selikhov A.V. Emergence and Propagation of Round Autowave in Cellular Neural Network // Lecture Notes in Computer Science. – 1999. – Vol. 1662. – P. 120–134.

[10] Beletkov D., Ostapkevich M., Piskunov S., Zhileev I. WinALT – Software tool for fine-grained algorithms and structure synthesis // Lecture Notes in Computer Science. – 1999. – Vol. 1662. – P. 491–496.