The Fourteenth International Symposium on Artificial Life and Robotics 2009 (AROB 14th '09),
B-Con Plaza, Beppu, Oita, Japan, February 5 - 7, 2009

Simulating the behaviour of cellular automata by extended spiking neural P systems

Aneta BINDER[1] Rudolf FREUND[1] Marion OSWALD[1,2]

[1]Faculty of Informatics
Vienna University of Technology
Vienna, Austria

[2]Computer and Automation Research Institute
Hungarian Academy of Sciences
Budapest, Hungary

ani@emcc.at rudi@emcc.at marion@emcc.at

Abstract

In cellular automata, the cells evolve depending on their own state and the states in their neighborhood, whereas the cells in extended spiking neural P systems act according to the number of spikes within the cells at each discrete time step. In this paper, we show how the looking for the states of the surrounding cells from a cell of a cellular automaton can be simulated by extended spiking neural P systems receiving specific numbers of spikes from the neighboring cells by using suitable encodings.

1 Introduction

Cellular automata are a well-established model for investigating and simulating the behavior of complex biological systems in discrete time steps and in a parallel way. Just recently, (extended) spiking neural P systems have been introduced (e.g., see [1]) arising from the idea of modeling the signal transmission between neuronal cells in the brain, e.g., see [2] as well as [6], [8] and [9]. Whereas the cells in cellular automata evolve depending on their own state and the states in their neighborhood, the cells in extended spiking neural P systems act according to the number of spikes within the cells at each discrete time step. Classic variants of cellular automata are based on a k-dimensional grid structure of the cells, e.g., we mention Conway's famous game of life (we also refer to [14] for a thorough discussion of two-dimensional cellular automata). Spiking neural P systems in principle were defined to work on arbitrary underlying graph structures, which more resembles the concept of generalized automata networks, e.g., see [3] and especially [13]. Extended spiking neural P systems are a special variant of membrane systems (see [10]; for the actual state of the art in this area, we refer to [15]). Inspired by the idea of signal transmission in the human brain, spikes are sent along the axons between cells (neurons), e.g., see [5] and especially [7] and [11]. The number of spikes sent to the surrounding cells depends on the actual contents (the number of spikes) in a specific cell. A spiking neural P system to be finite means that, at any moment, the number of spikes in any cell cannot exceed a given bound. The cells evolve in parallel at any time step, and depending on the actual number of spikes, each cell consumes some spikes and sends different numbers of spikes to its neighboring cells which are accumulated there to be considered in the next time step. Using suitable encodings, we can simulate the looking for the states of the surrounding cells from a cell of a cellular automaton by receiving specific numbers of spikes from these neighbor cells in the corresponding cell of the simulating extended spiking neural P system. In that way, the function and the behavior of a cellular automaton can exactly be simulated by a corresponding extended spiking neural P system, no matter which is the underlying connection structure. Extensions of the classic models of cellular automata can be simulated by suitable extended spiking neural P systems as well.

2 Definitions

For the basic elements of formal language theory needed in the following, we refer to any monograph in this area, in particular, to [12]. We just list a few notions and notations:

$\mathbb{N}$ denotes the set of non-negative integers. The interval of non-negative integers between k and m is denoted by $[k..m]$. For a finite set N, $|N|$ denotes the cardinality of N.

2.1 ESNP systems with total spiking

We here give a definition of a particular kind of extended spiking neural P systems, i.e., ESNP systems with total spiking (see [4]). For some original definitions we refer to [7] and [1].

An *extended spiking neural P system with total spiking* (*ESNP$_t$ system* for short) is a construct

$$\Pi = (V, S, R)$$

where

- V is a set of *cells* (or *neurons*); if V is a finite set, then the neurons are uniquely identified by a number between 1 and m;
- S describes the *initial configuration* by assigning an initial value (of spikes) to each cell;
- R is a set of *rules* of the form $(i, E/ \rightarrow P)$ such that $i \in V$ (specifying that this rule is assigned to cell i), E is a regular *checking set* (the current number of spikes in the cell has to be from E if this rule shall be executed), and P is a (possibly empty) set of *productions* of the form (l, w) where $l \in V$ (thus specifying the target cell), and $w \in \mathbb{N}$ is the *weight* of the energy (i.e., the number of spikes) sent along the axon from cell i to cell l.

Starting from the initial configuration given by S, a transition to the next configuration is performed by applying one rule $(i, E/ \rightarrow P)$ in each cell, if the current contents of the corresponding cell coincides with E. (Note that the whole contents of the neuron is lost as soon as a spiking rule $(i, E/ \rightarrow P)$ can be applied in neuron i). If there are more rules to be applied in one cell, then one is non-deterministically chosen. Hence, the system works in a sequential way on the level of the cells, but in a parallel way at the level of the whole system.

In the following, we will mainly consider *bounded* ESNP$_t$ systems, where for every cell, the number of cells from which it may receive an input is bounded and where at any moment, the number of spikes in it cannot exceed a bound specific for this cell. The cells evolve in parallel at any time step, and depending on the actual number of spikes, each cell consumes all of its spikes and sends different numbers of spikes to its neighboring cells which are accumulated there to be considered in the next time step. An ESNP$_t$ is called *finite*, if it is bounded and, moreover, V is finite.

2.2 Generalized automata networks and cellular automata

In the following, instead of the basic model of cellular automata (e.g., see [14]), we will consider a more general model based on the generalized automata networks (GAN) as described in [13].The main difference lies in the network topology: while the topology of CA is a d-dimensional lattice, GAN are built on an arbitrary directed graph.

Here, we define a GAN as a construct

$$A_G = (G, Q, f)$$

where

- $G = (V, E)$ is a directed graph, V is a set of vertices also called *cells*, and E is a set of edges,
- Q is a set of *states*,
- $f = \{f_i \mid i \in V\}$ is the *transition function*, where f_i is the local transition function of cell i which, depending on the states of the cells j with $(j, i) \in E$, determines the new state of cell i.

For a GAN A_G, a configuration at time t is defined as

$$C(t) = (q_1(t), q_2(t), ..., q_k(t)),$$

where $q_i(t) \in Q$ is the state of cell i at time t. The evolution of the GAN in time is then given by the iteration of the *evolution operator* $\Phi : C(t) \rightarrow C(t+1)$ for $t = 0, 1, ...$, through the simultaneous application in each cell of the local transition rule f.

Usually we assume the set of states Q to be finite, i.e., the cells can only choose from a bounded number of states, and, moreover, that the in-degree as well as the out-degree of every vertex (cell) in the graph G is finite, too; then such a GAN is called *bounded*. If, moreover, V is finite, too, then the GAN is called *finite*.

Example 1 *Consider the GAN $A_G = ((V, E), Q, f)$ with $V = \{1, ..., n\}$ for some $n \in \mathbb{N}$, $E = \{(i, i+1) \mid 1 \leq i < n\} \cup \{(n, 1)\}$, let Q be a finite set of states, f be the transition function with $f_i(q) = q$, and let S describe the* initial configuration *assigning an initial value from Q to each cell. The cells are connected in a simple ring structure by the edges in E,and the local transition function f_i just takes over the state q from*

cell $i+1$ to cell i (from cell n to cell 1). Hence, the time evolution of the GAN is periodic with $C(t) \to C(t+n)$ for $t = 0, 1, ...,$ with $C(0) = S$. The in-degree of every cell is 1, hence, with V and Q being finite, A_G is finite.

Obviously, in order to allow for a finite description, the graph structure as well as the local transition functions for the cells must follow specific restricted rules. The most typical example for a GAN with uniform rules is a (bounded, uniform) *cellular automaton* A_C (CA for short), which is defined on a d-dimensional grid, i.e., $V = \mathbb{N}^d$, $d \in \mathbb{N}$ is the dimension of A_C,

$$E = \{(i,i), (i+r_1, i), ..., (i+r_{n-1}, i) \mid i \in \mathbb{N}^d\},$$

where n is a fixed parameter determining the neighborhood size, and the r_j, $1 \leq j \leq n-1$, are fixed vectors in the d-dimensional space. Hence, instead of the graph $G = (V, E)$ we may simply define the so-called *neighborhood*

$$U = \{r_1, r_2, ..., r_{n-1}\}$$

and thus write a CA as

$$A_C = (d, Q, U, f)$$

where the *transition function* $f : Q^n \to Q$ maps the state of each cell $i \in \mathbb{N}^d$ to another state from Q as a function of the states of cell i and the cells in the neighborhood $U_i = \{i + r_1, i + r_2, ..., i + r_{n-1}\}$ of cell i. If Q – as it is usually assumed – is finite, then, by the definition given above, the CA A_C is bounded, but not finite. Obviously, there are also many variants of finite CA taking only a finite subspace of $\mathbb{N}^d$ as the underlying set of cells, but we do not go into such details here, as the main result established in the succeeding section holds true for such a variant just as a special case of a GAN.

Example 2 *Consider the CA $A_C = ((2, \{0,1\}, \{(1,1)\}, f)$ with $f(q) = q$ as in the first example. The uniform environment $\{(1,1)\}$ corresponds with*

$$E = \{((i,i), (i+1, i+1)) \mid i \in \mathbb{N}^2\},$$

hence, the CA A_C corresponds with the GAN $A_G = ((\mathbb{N}^2, E), \{0,1\}, f)$,which is bounded, but not finite. The local transition function f_i just takes over the state q from cell $(i+1, i+1)$ to cell (i,i). Hence, the time evolution of the CA given by the evolution operator $\Phi : C(t) \to C(t+1)$ for $t = 0, 1, ...,$ can be described by saying that the initial pattern given at time $t = 0$ is shifted one position to the right and one position up in every time step.

3 Results

We now present our main result, showing that the function and the behavior of a bounded generalized automata network or a cellular automaton can exactly be simulated by a corresponding extended spiking neural P system with total spiking (ESNP_t system), independent of the underlying connection structure of the GAN.

Using suitable encodings, we can simulate the looking for the states of the surrounding cells from a cell of a GAN by receiving specific numbers of spikes from these neighbor cells in the corresponding cell of the simulating ESNP_t system.

Theorem 1 *Any bounded generalized automata network (GAN) — independent of its underlying connection structure – can be simulated by a bounded extended spiking neural P system with total spiking (ESNP_t system). If the GAN is finite, then the ESNP_t system is finite, too.*

Proof. The GAN $A_G = ((V, E), Q, f)$ with the initial configuration S can be simulated by an ESNP_t system $\Pi = (V, S', R)$ in the following way:

First, we need an encoding function that allows us to encode all the information that may influence the state of a specific cell, i.e., we have to encode the state of the cell itself as well as the states of all cells from which it gets the information about their states due to the connections defined by E.

Now let n denote the cardinality of Q, i.e., n is the number of states used in A_G. These n states in Q can be ordered in a sequence $1, ..., n$, and each of these n numbers can be represented as a string representing the corresponding number between 1 and n in the dual system. Let $d(n)$ denote the number of digits needed to represent the number n in the dual system.

For a specific cell i, let the number of input cells for cell i be denoted by m_i and let these input cells j with $(j,i) \in E$ be written in a sequence $(j_{i,1}, ...j_{i,m_i})$. The current states of all these m_i input cells of cell i now have to be represented by a number of spikes which uniquely determines this specific situation from another one with these cells of Π being in other states. One specific way to represent the states of these m_i input cells $(j_{i,1}, ...j_{i,m_i})$ of cell i now is to concatenate the $\{0,1\}$-strings representing the state of each by $d(n)$ digits (i.e., with leading zeroes to have a string with exactly $d(n)$ digits for each cell $j_{i,l}$, $1 \leq l \leq m_i$). This string (of exactly $d(n) * m_i$ digits 0 or 1) corresponds with a number that in the ESNP_t system Π is the number of spikes representing the states of all its input cells in a cell i.

Now let us start with the initial configuration S' which in each cell i contains exactly that number of spikes which represents the state of cell i in S. The ESNP$_t$ system Π evolves simulating the transitions in the given GAN A_G by suitable rules in R for each cell i implementing the local transition function f_i: let y be the number of spikes in cell i and $x = y/2^{d(n)m_i}$; for $f_i(x_1, ..., x_{m_i})$ and with x being interpreted as the number encoding the states $x_1, ..., x_{m_i}$ as described above we use $(i, \{x\}/ \rightarrow P)$ with P being the set of productions containing the production $\big(i, f_i(x_1, ..., x_{m_i})2^{d(n)m_i}\big)$ – sending the information about its new state to cell i itself – as well as $\big(j, f_i(x_1, ..., x_{m_i})2^{d(n)(p(i,j)-1)}\big)$ for every cell j which has to receive the information about the state of cell i, i.e., with $(i, j) \in E$; $p(i, j)$ denotes the position of cell i in the sequence of input cells for cell j. In that way, the information about the new states of the cells in the given GAN A_G is propagated as the corresponding number of spikes in the ESNP$_t$ system Π. At any moment, in each cell i of the ESNP$_t$ system Π the corresponding state of the underlying GAN can be recovered from the number of spikes y by dividing y by $2^{d(n)m_i}$.

Obviously, if the underlying GAN A_G is finite, then by the construction of the ESNP$_t$ system Π given above, Π is finite, too. □

As a special consequence of the preceding theorem we obtain the result that the evolution of cellular automata can be simulated by ESNP$_t$ systems.

4 Conclusion

We have shown how cellular automata or even generalized automata networks can be simulated by ESNP$_t$ systems, where the states of the cells are represented by the corresponding number of spikes.

It is an open question how such a simulation could also be done by other variants of extended spiking neural P systems as, for example, by ESNP systems with decay or thresholds (see [4]).

References

[1] Alhazov A, Freund R, Oswald M, Slavkovik M (2007) Extended spiking neural P systems generating strings and vectors of non-negative integers, in Hoogeboom HJ, Păun Gh, Rozenberg G (eds.), Membrane Computing, 7th International Workshop, WMC7, Springer, Berlin, 123–134

[2] Buarque de Lima Neto F, de Wilde P (2006) Venn-like models of neo-cortex patches, International Joint Conference on Neural Networks, 89–96.

[3] Csuhaj-Varjú E (2001) Networks of language processors, Current Trends in Theoretical Computer Science. World Scientific, 771–790

[4] Freund R, Ionescu M, Oswald M (2008) Extended spiking neural P systems with decaying spikes and/or total spiking, Intern J Found Computer Sci 19(5), 1223–1234

[5] Martín-Vide C, Pazos J, Păun Gh, Rodríguez-Patón A (2002) A new class of symbolic abstract neural nets: Tissue P systems, in Proceedings of COCOON 2002, Singapore, Lecture Notes in Computer Science 2387, Springer-Verlag, Berlin, 290–299

[6] Gerstner W, Kistler W (2002) Spiking Neuron Models. Single Neurons, Populations, Plasticity. Cambridge Univ. Press

[7] Ionescu M, Păun Gh, Yokomori T (2006) Spiking neural P systems, Fundamenta Informaticae 71, 2–3:279–308

[8] Maass W (2002) Computing with spikes. Special Issue on Foundations of Information Processing of TELEMATIK 8, 1:32–36

[9] Maass W, Bishop C (eds) (1999) Pulsed Neural Networks. MIT Press, Cambridge

[10] Păun Gh (2002) Computing with Membranes: An Introduction. Springer, Berlin

[11] Păun Gh, Pérez-Jiménez MJ, Rozenberg G (2006) Spike trains in spiking neural P systems, Intern J Found Computer Sci 17(4), 975–1002

[12] Rozenberg G, Salomaa A (eds) (1997) Handbook of Formal Languages (3 volumes). Springer, Berlin

[13] Tomassini M (2006), Generalized Automata Networks, Yacoubi S El, Chopard B, Bandini S (Eds.), ACRI 2006, Lecture Notes in Computer Science 4173, Springer, Berlin, 14–28

[14] Wolfram S (2002), A new kind of science, Wolfram Media

[15] The P Systems Web Page, `http://ppage.psystems.eu`