13th International Conference on Membrane Computing, CMC13,
Budapest, Hungary, August 28 - 31, 2012. Proceedings, pages 259 - 275.

Spiking Neural P Systems with Functional Astrocytes

Luis F. Macías-Ramos, Mario J. Pérez-Jiménez

Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla, Spain
Avda. Reina Mercedes s/n. 41012 Sevilla, Spain
lfmaciasr@us.es, marper@us.es

Abstract. Spiking Neural P Systems (SN P Systems, for short) is a developing field within the universe of P Systems. New variants arise constantly as the study of their properties, such as computational completeness and computational efficiency, grows. Variants frequently incorporate new ingredients into the original model inspired by real neurophysiological structure of the brain. A singular element present within that structure is the astrocyte. Astrocytes, also known collectively as astroglia, are characteristic star-shaped glial cells in the brain and spinal cord. In this paper, a new variant of Spiking Neural P Systems incorporating astrocytes is introduced. These astrocytes are modelled as computing devices capable of performing function computation in a single computation step. In order to experimentally study the action of Spiking Neural P Systems with astrocytes, it is necessary to develop software providing the required simulation tools. Within this trend, P–Lingua offers a standard language for the definition of P Systems. Part of the same software project, *pLinguaCore* library provides particular implementations of parsers and simulators for the models specified in P–Lingua. Along with the new SN P System variant with astrocytes, an extension of the P–Lingua language allowing definition of these systems is presented in this paper, as well as an upgrade of *pLinguaCore*, including a parser and a simulator that supports the aforementioned variant.

1 Introduction

Spiking Neural P Systems were introduced in [10] in the framework of membrane computing [16] as a new class of computing devices which are inspired by the neurophysiological behaviour of neurons sending electrical impulses (spikes) along axons to other neurons.

A SN P System consists of a set of neurons placed as nodes of a directed graph (called the *synapse graph*). Each neuron contains a number of copies of a single object type, the *spike*. Rules are assigned to neurons to control the way information flows between connected neurons, i.e. rules assigned to a neuron allow it to send spikes to its neighbouring neurons. SN P Systems usually work

in a synchronous mode, where a global clock is assumed. In each time unit, for each neuron, only one of the applicable rules is non-deterministically selected to be executed. Execution of rules takes place in parallel amongst all neurons of the system.

Since the introduction of this model, many computational properties of SN P Systems have been studied. It has been proved that they are **Turing**-complete when considered as number computing devices [10], used as language generators [5,3], or computing functions [15]. Also, many variants have come into scene bringing new ingredients into the model (or sometimes dropping some of them), while others modify its behaviour, that is, its semantics. Motivation of this "research boom" can be found in a quest for both enhancing expressivity and efficiency of the model, as well as exploring its computational power.

As a direct result of all of this, there is an extensive (and growing) bibliography related to SN P Systems. For instance, it has been shown [4] how usage of pre-computed resources makes them able to solve computationally hard problems in constant time. Also, study of different kinds of asynchronous "working modes" has been conducted [18]. In what concerns to the addition of new ingredients into the model, this involves (naming only some examples) weights [20], *antispikes* [12], extended rules [18] or budding and division rules [13].

A SN P Systems variant with astrocytes was first introduced in [2]. Astrocytes are glial cells connected to one or more synapses that can sense the whole spike traffic passing along their neighbouring synapses and, eventually, modify it. Their functionalities include biochemical support of endothelial cells that form the blood-brain barrier, provision of nutrients to the nervous tissue, maintenance of extracellular ion balance, and a role in the repair and scarring process of the brain and spinal cord following traumatic injuries. It has been shown that astrocytes propagate intercellular Ca_2^+ waves over long distances in response to stimulation, and, similarly to neurons, release transmitters (called gliotransmitters) in a Ca_2^+-dependent manner.

Moreover, within the dorsal horn of the spinal cord, activated astrocytes have the ability to respond to almost all neurotransmitters [9] and, upon activation, release a multitude of neuroactive molecules that influences neuronal excitability. Synaptic modulation by astrocytes takes place because of the 3-part association between astrocytes and presynaptic and postsynaptic terminals forming the so-called "tripartite synapse" [1].

Such discoveries have made astrocytes an important area of research within the field of neuroscience, thus also an interesting element to consider bringing into Natural Computing disciplines like Membrane Computing.

The model presented in [2], pretty complex, was then simplified in [17], in which only inhibitory astrocytes were considered. This simplification was recently revised again in [14], where "hybrid" astrocytes were introduced. Behaviour of an astrocyte of this kind, inhibitory or excitatory, relied on the amount of spikes passing on its neighbouring synapses, in relation to a given threshold associated to it. Thus, for a given astrocyte *ast* with associated threshold t with k spikes passing along its neighbouring synapses syn_{ast} at a certain instant, a) if $k > t$, the astrocyte *ast* has an inhibitory influence on the neighbouring synapses, and the k spikes are simultaneously suppressed (that is, the spikes are removed from the system); b) if $k < t$, the astrocyte *ast* has an excitatory influence on the neighbouring synapses, all spikes survive and pass to their destination neurons, reaching them simultaneously; c) if $k = t$, the astrocyte *ast* non-deterministically chooses an inhibitory or excitatory influence on the neighbouring synapses. It is possible for two or more astrocytes to control the same synapse. In this case, only if every astrocyte has an excitatory influence on the synapse the spikes passing along that synapse survive.

In this paper, again, a new variant is introduced. Based upon the original model defined in [2], new ingredients are introduced in order to turn astrocytes into *function computation devices*. Briefly, a set of pairs (threshold, function) is associated with each astrocyte. Existing spike traffic measured on distinguished neighbouring *control synapses* attached to the astrocyte is matched against the thresholds until one of them is selected. Subsequently, the associated function to the matched threshold is selected. At this point, that function is computed taking as arguments the amounts of spikes measured on distinguished neighbouring *operand synapses* attached to the astrocyte. Finally, the result of the function computation is sent through a distinguished operand synapse.

So, by introducing this new kind of astrocytes, not only covering of functionality of the astrocytes defined in [2] is achieved, also any computable partial function between natural numbers can be computed in a single computation step. Moreover, this new ingredient eases the design of machines that calculate functions, as astrocytes can be viewed as "macros".

In addition, a P–Lingua based simulator for the proposed model has been developed, which also simulates the model defined in [14]. The aforementioned simulator is an extension of the one presented in [11]. P–Lingua is a programming language intended to define P Systems [7,8,19], that comes together with a Java library providing several services (e.g., parsers for input files and built-in simulators).

This paper is structured as follows. Section 2 is devoted to introduce the formal specification of SN P Systems with Functional Astrocytes (*SNPSFA* for short). Section 3, is devoted to show applications of the presented model. Section 4 is devoted to simulation: A P–Lingua syntax for defining SNPSFA

is introduced, along with several examples. Finally, the simulation algorithm is shown. Section 5 covers conclusions and future work.

2 Spiking Neural P Systems with Functional Astrocytes

In this section, we introduce SN P Systems with Functional Astrocytes.

2.1 Syntax

A *Spiking Neural P System with Functional Astrocytes* (SNPSFA for short) of degree $(m, l), m \geq 1, l \geq 1$, is a construct of the form

$$\Pi = (O, \sigma, syn, ast, out), \text{ where:}$$

- $O = \{a\}$ is the singleton alphabet (a is called *spike*);
- $\sigma = \{\sigma_1, \ldots, \sigma_m\}$ is the finite set of neurons, of the form $\sigma_i = (n_i, R_i), 1 \leq i \leq m$, where:
  - $n_i \geq 0$ is the initial number of spikes contained in σ_i;
  - R_i is a finite set of extended rules of the following form:

$$E/a^c \rightarrow a^p$$

    where E is a regular expression over a, and $c \geq 1$, $p \geq 1$ with $c \geq p$;
- $syn = \{s_1, \ldots, s_\theta\} \subseteq \{1, \ldots, m\} \times \{1, \ldots, m\}$ with $(i, i) \notin syn$ is the *set of synapses*;
- $ast = \{ast_1, \ldots, ast_l\}$ is the finite set of astrocytes, with $ast_j, (1 \leq j \leq l)$ of the form

$$ast_j = (syn_j^o, syn_j^c, \omega_j, T_j, F_j, p_j(0), \gamma_j), \text{ where:}$$

  - $syn_j^o = \{s_{j,1}^o, \ldots, s_{j,r_j}^o\} \subseteq syn, r_j \geq 1$, is the astrocyte finite set of *operand synapses*, ordered by a lexicographical order imposed on syn_j^o;
  - $syn_j^c = \{s_{j,1}^c, \ldots, s_{j,q_j}^c\} \subseteq syn, q_j \geq 0$, is the astrocyte finite set of *control synapses*;
  - $\omega_j \in \{true, false\}$ is the astrocyte control-as-operand flag;
  - $T_j = \{T_{j,1}, \ldots, T_{j,k_j}\}, k_j \geq 1$, is the astrocyte finite set of thresholds, such that, $T_{j,\alpha} \in \mathbb{N}$, $(1 \leq \alpha \leq k_j)$ and $T_{j,1} < \ldots < T_{j,k_j}$;
  - $F_j = \{f_{j,1}, \ldots, f_{j,k_j}\}$ is the astrocyte finite *multiset* (some elements in F_j can be the same) of natural functions such that for each α $(1 \leq \alpha \leq k_j)$:
    - $f_{j,\alpha}$ is a computable function between natural numbers;
    - if $\omega_j = true$ then $f_{j,\alpha}$ is a unary function;
    - if $\omega_j = false$ and $r_j = 1$ then $f_{j,\alpha}$ is a unary constant function;
    - if $\omega_j = false$ and $r_j > 1$ then $f_{j,\alpha}$ has arity $r_j - 1$;
  - $p_j(0) \in \mathbb{N}$ is the astrocyte initial potential;
  - $\gamma_j \in \{true, false\}$ is the astrocyte potential update flag;
- $out \in \sigma$ is the output neuron.

2.2 Semantics

In order to precise semantics of a SNPSFA, let us informally introduce some topological aspects of the model and the nature of the firing process. Given a synapse $s_g = (\sigma_{g,1}, \sigma_{g,2}) \in syn$, if an astrocyte is linked to s_g, it can be viewed as that it "makes contact" with s_g in the "space between" s_g^1 and s_g^2 (it can be said that the astrocyte is "attached" to the synapse as well). If there exists several astrocytes attached to s_g, all of them make contact at the same intermediate point. These astrocytes can simultaneously read the spike traffic going from $\sigma_{g,1}$ to $\sigma_{g,2}$ at an instant t and eventually modify it.

Keeping in mind the intuitive ideas expressed above, we proceed now to formally specify the semantics of *SN P Systems with Functional Astrocytes* as an extension of the one defined for the well-known SN P Systems model. A global clock is assumed and in every computation step one and only one rule can be selected for a given neuron. Let us introduce the following notation as a matter of convenience: given a synapse $s_y = (\sigma_y^1, \sigma_y^2)$, we denote by σ_y^1 the input neuron of s_y and by σ_y^2 the output neuron of s_y.

An astrocyte can sense the spike traffic passing along its neighbouring synapses, both control and operand ones. For an astrocyte ast_j, if there are k spikes passing along the control synapses in an instant t and the current potential of ast_j at t is p, then the value $s = k + p$ is computed. At this point, the number h satisfying that $s \in [T_{j,h}, T_{j,h+1})$ is computed out of s. Let us notice that if $s < T_{j,1}$ then $h = 1$, and if $s > T_{j,k_j}$ then $h = k_j$. Following this, by using both h and the boolean value ω_j, a number s' is computed as follows. If $\omega_j = true$ then $s' = f_{j,h}(s)$ directly. Otherwise, two cases are considered: a) if the number of operand synapses r_j is one, then $s' = f_{j,h}(0)$; and b) if the number of operand synapses is greater than one and assuming that $x_1, x_2, \ldots, x_{r_j-1}$ spikes are passing along the respective operand synapses associated to ast_j, then $s' = f_{j,h}(x_1, x_2, \ldots, x_{r_j-1})$. Finally, the multisets of the input and output neurons associated to the operand and control synapses are updated. For the output neurons: a) if they are associated to control synapses, then their corresponding multisets are added the spikes passing along the synapses at instant t; and b) if they are associated to operand synapses, then no change is applied to their multisets, except for neuron s_{j,r_j}^o, which is added s' spikes. Similarly, multisets corresponding to input neurons associated to both operand and control synapses are subtracted the spikes passing along the aforementioned synapses at instant t.

As a last remark, if the astrocyte potential update flag $\gamma_j = true$ then the astrocyte potential in $t + 1$ will be incremented in s units. Otherwise, the astrocyte potential does not change.

3 Applications of Spiking Neural P Systems with Functional Astrocytes

As mentioned before, by introducing SNPSFA covering of functionality of astrocytes defined in [2] is achieved. Also, astrocytes within SNPSFA are able to compute any computable natural partial function $f : \mathbb{N}^m - \rightarrow \mathbb{N}$ in a single computation step. Let us illustrate this fact by showing how to re-implement the examples covered in [2] within the scope of our proposed model. Moreover, the corresponding P–Lingua files for the aforementioned examples are covered in Section 4, thus by running the introduced simulator against these files, its working process can be checked in relation to the semantics presented above.

3.1 Excitatory and Inhibitory Astrocytes

First couple of examples shows how to implement excitatory and inhibitory astrocytes respectively, with a given threshold k. Implementation involves defining two functions: $f(x)$, which is the identically zero function of arity one, and $g(x)$.

Excitatory astrocyte, ast_{exc}, is depicted in the Fig. 1 with its formal specification being:

$$ast_{exc} = (\{(p', q)\}, \{(p, q')\}, true, \{0, k\}, \{f(x), g(x)\}, 0, false)$$

and its working equation, assuming that α spikes pass through synapse (p, q') at a given instant t, being:

$$ast_{exc}(\alpha, t) = \begin{cases} f(\alpha) = 0 & \text{if } 0 \leq \alpha < k \\ g(\alpha) & \text{if } \alpha \geq k \end{cases}$$

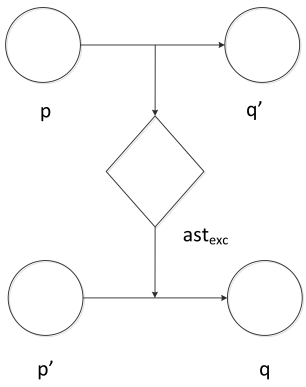

Fig. 1. Excitatory astrocyte.

Inhibitory astrocyte, ast_{inh}, is structurally identical to ast_{exc}, with its formal specification being:

$$ast_{inh} = (\{(p', q)\}, \{(p, q')\}, true, \{0, k+1\}, \{g(x), f(x)\}, 0, false)$$

and its working equation, assuming that α spikes pass through synapse (p, q') at a given instant t, being:

$$ast_{inh}(\alpha, t) = \begin{cases} g(\alpha) & \text{if } 0 \leq \alpha \leq k \\ f(\alpha) = 0 & \text{if } \alpha \geq k+1 \end{cases}$$

3.2 Logic Gates

Second couple of examples shows how to implement logical gates, concretely AND-gates and NAND-gates respectively. Implementation involves defining two functions, $f(x)$ and $g(x)$, both of them unary constant functions, which associates the 0 and 1 natural values respectively for every $x \in \mathbb{N}$.

AND-gate astrocyte, ast_{and}, is depicted in the Fig. 2 with its formal specification being:

$$ast_{and} = (\{(p, q)\}, \{(A, A'), (B, B')\}, false, \{1, 2\}, \{f(x), g(x)\}, 0, false)$$

and its working equation, assuming that $\alpha, 0 \leq \alpha \leq 2$ spikes in total pass through synapses (A, A') and (B, B') at a given instant t, being:

$$ast_{and}(\alpha, t) = \begin{cases} f(0) = 0 & \text{if } 0 \leq \alpha \leq 1 \\ g(0) = 1 & \text{if } \alpha = 2 \end{cases}$$

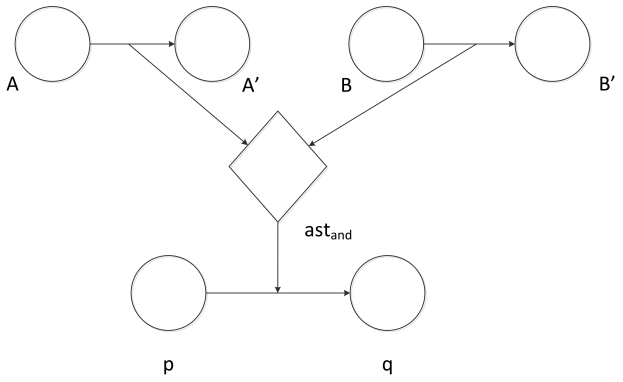

Fig. 2. AND-gate astrocyte.

NAND-gate astrocyte, ast_{nand}, is structurally identical to ast_{and}, with its formal specification being:

$$ast_{nand} = (\{(p,q)\}, \{(A,A'),(B,B')\}, false, \{1,2\}, \{g(x), f(x)\}, 0, false)$$

and its working equation, assuming that $\alpha, 0 \leq \alpha \leq 2$ spikes in total pass through synapses (A,A') and (B,B') at a given instant t, being:

$$ast_{nand}(\alpha, t) = \begin{cases} g(0) = 1 & \text{if } 0 \leq \alpha \leq 1 \\ f(0) = 0 & \text{if } \alpha = 2 \end{cases}$$

3.3 Discrete Amplifier

Last example shows how to implement a discrete amplifier which, as soon as the spike amount passing through control synapse (B,B') goes beyond a given threshold k, computes the amplification function $f_{*,n}(x) = n * x$ from the input given at E, otherwise no amplification is performed. Rules $a^l \rightarrow a^l$ belonging to neuron p are interpreted in the same way as in [2]. Implementation involves defining two functions: $g(x) = f_{*,n}(x)$ and $f(x)$, which associates x for every $x \in \mathbb{N}$.

Discrete amplifier astrocyte, ast_{amp}, is depicted in the Fig. 3 with its formal specification being:

$$ast_{amp} = (\{(p,p'),(q',q)\}, \{(B,B')\}, false, \{0,k\}, \{f(x), g(x)\}, 0, false)$$

and its working equation, assuming that at a given instant t α spikes pass through synapse (B,B') and β spikes pass through synapse (p,p'), being:

$$ast_{amp}(\alpha, \beta, t) = \begin{cases} f(\beta) = \beta & \text{if } 0 \leq \alpha < k \\ g(\beta) = n * \beta & \text{if } \alpha \geq k \end{cases}$$

4 A P–Lingua Based Simulator for SNPSFA

This section introduces a P–Lingua simulator for SNPSFA, extending the one presented in [11]. *SNPSFA are only partially simulated because only certain functions can be defined within P–Lingua framework.* Also, let us notice that an extension of the simulator presented here intended to simulate SNPSA as introduced in [14] is being developed.

P–Lingua syntax for specifying aforementioned SNPSFA is introduced, along with several examples. To conclude, the simulation algorithm is shown.

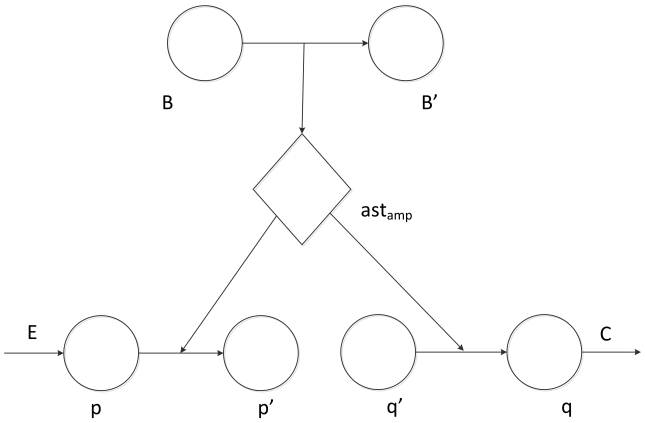

Fig. 3. Discrete amplifier astrocyte.

4.1 P–Lingua Syntax

A set of new features has been incorporated into P–Lingua in order to support SNPSFA. New instructions have been included to define both astrocytes and functions, extending the P–Lingua model specification framework for Spiking Neural P Systems. Thus, these instructions can be used only when the source P–Lingua files defining the models begin with the following sentence:

```
@model<spiking_psystems>
```

In what follows, P–Lingua syntax for defining SNPSFA is introduced.

– *Astrocytes.*

The following sentence can be used to define a SNPSFA astrocyte ast_j^b, with b standing for *binder*, as the astrocytes presented in [2] inspired the functional astrocytes presented in this paper:

```
@mastb =
(
   label-j,
   operand-synapses-j,control-synapses-j,control-operand-flag-j,
   set-thresholds-j,set-functions-j,
   potential-j,update-potential-j
);
```

where:

- *label-j* is the label of the astrocyte;

- *operand-synapses-j* is the set of operand synapses associated to the astrocyte, with *operand-synapses-j* $= \{s^o_{j,1}, \ldots, s^o_{j,r_j}\}$ and $s^o_{j,v} = (\sigma^{o,1}_{j,v}, \sigma^{o,2}_{j,v})$, a pair of neuron labels defining the synapse;

- *control-synapses-j* is the set of control synapses associated to the astrocyte, with *control-synapses-j* $= \{s^c_{j,1}, \ldots, s^c_{j,q_j}\}$ and $s^c_{j,u} = (\sigma^{c,1}_{j,u}, \sigma^{c,2}_{j,u})$, a pair of neuron labels defining the synapse;

- *control-operand-flag-j* is the astrocyte control-as-operand flag, with *control-operand-flag-j* $\in \{true, false\}$;

- *set-thresholds-j* is the astrocyte natural set of thresholds, defined as *set-thresholds-j* $= \{T_{j,1}, \ldots, T_{j,k_j}\}$ with $T_{j,1} < \ldots < T_{j,k_j}$;

- *set-functions-j* is the astrocyte set of natural computable functions, defined as *set-functions-j* $= \{f_{j,1}, \ldots, f_{j,k_j}\}$, all of them having the same arity;

- *potential-j* is the astrocyte initial potential, with *potential-j* $\in \mathbb{N}$;

- *update-potential-j* is the astrocyte potential update flag, verifying that *update-potential-j* $\in \{true, false\}$;

– *Functions.*

The following sentence can be used to define a function of name *f-name*:

```
@mastfunc =
(
   f-name(x1,...,xN),
   f-name(x1,...,xN) = "expr(x1,...,xN)"
);
```

where:

- *f-name* is the function name, a P–Lingua identifier;

- $x1, \ldots, xN$ is the list of arguments; notation for naming arguments must follow the convention of starting with x and immediately being followed by a integer literal, starting with 1 and being incremented in one unit each time;

- $exp(x1, ..., xN)$ is the function defining expression; this expression must yield a natural number; because of the underlying coding library, *exp4j* [6], definition of functions is restricted to use elements shown at

`http://projects.congrace.de/exp4j/`;

Let us notice that, as we are restricted when defining functions, SNPSFA are only partially simulated. The following functions are pre-defined, thus can be used directly, without having to be explicitly defined in the P–Lingua source file:

- $zero(x1)$ is the identically zero function of arity one;
- $identity(x1)$ is the identity function of arity one;
- $pol()$ is a *function template* allowing the definition of a polynomial astrocyte function $pol(x_0, x_1, \ldots, x_n, x)$ of any arity $n + 2, n \geq 0$, defined as follows:

$$pol(x_0, x_1, \ldots, x_n, x) = x_0 + \sum_{i=1}^{n} x_i * x^i$$

with $x_i \in \mathbb{N}, 0 \leq i \leq n, x \in \mathbb{N}$;

$x_0, \ldots, x_n, x$ arguments take value from the spikes passing through the operand synapses associated to a given astrocyte ast_j at a instant t in the following way:

$$\begin{cases} x_0 \leftarrow s^o_{j,1}(t) \\ x_1 \leftarrow s^o_{j,2}(t) \\ \ldots \\ x_n \leftarrow s^o_{j,r_{j-2}}(t) \\ x \leftarrow s^o_{j,r_{j-1}}(t) \end{cases}$$

- $sub()$ is a *function template* allowing the definition of a natural substraction function $sub(x_1, \ldots, x_n)$ of any arity n greater or equal than one, defined as follows:

$$sub(x_1, \ldots, x_n) = \begin{cases} x_1 - x_2 - \cdots - x_n & \text{when} \quad x_1 - x_2 - \cdots - x_n \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

with $x_i \in \mathbb{N}, 1 \leq i \leq n$;

$x_1, \ldots, x_n$ arguments take value from the spikes passing through the operand synapses of a given astrocyte ast_j at an instant t in the following way:

$$\begin{cases} x_1 \leftarrow s^o_{j,1}(t) \\ \ldots \\ x_n \leftarrow s^o_{j,r_{j-1}}(t) \end{cases}$$

Let us notice that if $n = 1$ and the astrocyte control-as-operand-flag is set, then it is trivial to show that $sub(x_1, \ldots, x_n) = potential(j, t) + spikes(j, t)$.

4.2 Examples

In what follows, a set of on line examples are listed. Each of them corresponds to a P–Lingua file that shows one of the SNPSFA applications presented in Section 3.

- Excitatory astrocyte:
  `http://www.p-lingua.org/examples/SNPSFA_excitatory.pli`.

- Inhibitory astrocyte:
  `http://www.p-lingua.org/examples/SNPSFA_inbitory.pli`.

- AND-gate:
  `http://www.p-lingua.org/examples/SNPSFA_AND_gate.pli`.

  For this example, forgetting rules have been used assuming a natural extension of the proposed model. This allows generating "random" boolean signals coming from neurons A and B.

- Discrete amplifier:
  `http://www.p-lingua.org/examples/SNPSFA_amplifier.pli`.

4.3 Simulation Algorithm

In [8], a Java library called *pLinguaCore* was presented under GPL license. The library provides parsers to handle input files, built–in simulators to generate P System computations and is able to export several output file formats that represent P Systems. *pLinguaCore* is not a closed product because developers with knowledge of Java can add new components to the library, thus extending it.

In this paper, an upgrade of the library is presented. Support for SNPSFA has been included, as an extension of the works presented in [11]. As a result of this, *pLinguaCore* is now able to handle input P–Lingua files defining SNPSFA. In addition, a new built–in simulator capable of simulating computations of these systems has been included into the library. For downloading the latest version of *pLinguaCore*, please refer to `http://www.p-lingua.org`. Also, a simulator for astrocytes as introduced in [14] is in development.

The following pseudo-code shows a computation step from instant t to $t+1$ for a SNPSFA, illustrating the way in which the simulator operates. The pseudo-code is structured in two algorithms, following the semantics of SNPSFA introduced in Section 2. The first one deals with the input neurons of the systems, while the second one deals with astrocytes and output neurons. Notation follows from Section 2, while additional required notation can be found at the end of this Section.

Algorithm 1 Neurons loop

1: let $\sigma = \{\sigma_1, \ldots, \sigma_m\}$ be the set of all the neurons in the system
2: **for** $i = 1$ **to** m **do**
3: $\quad \sigma_i(t+1) \leftarrow \sigma_i(t) - l_i(t)$
4: **end for**

Algorithm 2 Astrocytes loop

1: let $ast = \{ast_1, \ldots, ast_l\}$ be the set of all the astrocytes in the system
2: **for** $j = 1$ **to** l **do**
3: $\quad spikes(j,t) \leftarrow \sum_{u=1}^{q_j} s^c_{j,u}(t)$
4: $\quad selector(j,t) \leftarrow spikes(j,t) + p_j(t)$
5: $\quad h \leftarrow \begin{cases} 1 & \text{if } selector(j,t) < T_{j,1} \\ k_j & \text{if } selector(j,t) > T_{j,k_j} \\ e & \text{if } e = \max\{x \mid 1 \leq x \leq k_j \wedge T_{j,x} \leq selector(j,t)\} \end{cases}$
6: $\quad f^*_j \leftarrow f_{j,h}$
7: $\quad$ **if** $\omega_j = true$ **then**
8: $\quad\quad output(j,t) \leftarrow f^*_j(selector(j,t))$
9: $\quad$ **end if**
10: $\quad$ **if** $\omega_j = false$ and $r_j = 1$ **then**
11: $\quad\quad output(j,t) \leftarrow f^*_j(0)$
12: $\quad$ **end if**
13: $\quad$ **if** $\omega_j = false$ and $r_j > 1$ **then**
14: $\quad\quad output(j,t) \leftarrow f^*_j(s^o_{j,1}(t), \ldots, s^o_{j,r_j-1}(t))$
15: $\quad$ **end if**
16: $\quad$ **for** $u = 1$ **to** q_j **do**
17: $\quad\quad \sigma^{c,2}_{j,u}(t+1) \leftarrow \sigma^{c,2}_{j,u}(t) + s^c_{j,u}(t)$
18: $\quad$ **end for**
19: $\quad$ **for** $v = 1$ **to** $r_j - 1$ **do**
20: $\quad\quad \sigma^{o,2}_{j,v}(t+1) \leftarrow \sigma^{o,2}_{j,v}(t)$
21: $\quad$ **end for**
22: $\quad \sigma^{o,2}_{j,r_j}(t+1) \leftarrow \sigma^{o,2}_{j,r_j}(t) + output(j,t)$
23: $\quad$ **if** $\gamma_j = true$ **then**
24: $\quad\quad p_j(t+1) \leftarrow spikes(j,t)$
25: $\quad$ **end if**
26: **end for**

Required Additional Notation Following notation from Section 2, we introduce the following required additional notation.

– Given an astrocyte $ast_j, (1 \leq j \leq l)$, we denote the synapses attached to ast_j as $s^\lambda_{j,w} = (\sigma^{\lambda,1}_{j,w}, \sigma^{\lambda,2}_{j,w}), \lambda \in \{o,c\}, w \in \{u,v\}$, and:

- we denote the operand synapses of ast_j as

$$s^o_{j,v} = (\sigma^{o,1}_{j,v}, \sigma^{o,2}_{j,v}), 1 \leq v \leq r_j, (r_j \geq 1);$$

- we denote the control synapses of ast_j as

$$s^c_{j,u} = (\sigma^{c,1}_{j,u}, \sigma^{c,2}_{j,u}), 1 \leq u \leq q_j, (q_j \geq 0).$$

– Given an astrocyte $ast_j, (1 \leq j \leq l)$ attached to a synapse $s^{\lambda}_{j,w} = (\sigma^{\lambda,1}_{j,w}, \sigma^{\lambda,2}_{j,w})$ as defined above, we denote $s^{\lambda}_{j,w}(t)$ as the number of spikes fired by $\sigma^{\lambda,1}_{j,w}$ at an instant t of a computation.

– Given a neuron $\sigma_i, 1 \leq i \leq m$, we denote

  - $\sigma_i(t)$ = number of spikes contained in σ_i at instant t by a computation
  - $l_i(t)$ = number of spikes corresponding to the left hand side of the selected rule in neuron σ_i at instant t by a computation
  - $r_i(t)$ = number of spikes corresponding to the right hand side of the selected rule in neuron σ_i at instant t by a computation

5 Conclusions and Future Work

In this paper we present a new variant of Spiking Neural P Systems, wich includes astrocytes capable of calculating computable functions in a simple computation step. Applications of this variant are vast, as exemplified in the study cases shown, but yet to explore. In this sense, a new release of P–Lingua, that extends the previous SN P System simulator has been developed, incorporating the ability to work with astrocytes. This new simulator has been included into the library *pLinguaCore* and tested by simulating examples taken from the literature, concretely the ones existing in [14] and [2] (these ones adapted to the introduced SNPSFA variant).

At the moment, an extension of the implemented simulator supporting Spiking Neural P System with "hybrid" Astrocytes as defined in [14] is in development. Once this work is done, a desirable feature would be to provide a mechanism for defining arbitrary computable functions, thus fully simulating SNPSFA. Additional elements such as weights and antispikes might also be incorporated.

Acknowledgements

The authors acknowledge the support of the project TIN2009–13192 of the Ministerio de Ciencia e Innovación of Spain, cofinanced by FEDER funds, and the support of the Project of Excellence with *Investigador de Reconocida Valía* of the Junta de Andalucía, grant P08-TIC-04200.

References

1. Araque, A., Parpura, V., Sanzgiri, R.P., Haydon, P.G.: Tripartite synapses: glia, the unacknowledged partner. Trends in Neuroscience 22(5), 208–215 (1999)

2. Binder, A., Freund, R., Oswald, M., Vock, L.: Extended spiking neural P Systems with excitatory and inhibitory astrocytes. In: Proceedings of the 8th Conference on 8th WSEAS International Conference on Evolutionary Computing - Volume 8. pp. 320–325. EC'07, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA (2007), http://dl.acm.org/citation.cfm?id=1347992.1348008
3. Chen, H., Freund, R., Ionescu, M., Păun, G., Pérez-Jiménez, M.J.: On string languages generated by spiking neural P Systems. Fundam. Inform. 75(1-4), 141–162 (2007)
4. Chen, H., Ionescu, M., Isdorj, T.O.: On the efficiency of spiking neural P Systems. In: Proceedings of the 8th International Conference on Electronics, Information, and Communication, Ulanbator, Mongolia. pp. 49–52 (06 2006)
5. Chen, H., Ionescu, M., Ishdorj, T.O., Păun, A., Păun, G., Pérez-Jiménez, M.J.: Spiking neural P Systems with extended rules: universality and languages. Natural Computing 7(2), 147–166 (2008)
6. Congrace Developer Team: The exp4j website. http://projects.congrace.de/exp4j/
7. Díaz-Pernil, D., Pérez-Hurtado, I., Pérez-Jiménez, M.J., Riscos-Núñez, A.: A p-lingua programming environment for membrane computing. In: Corne, D.W., Frisco, P., Păun, G., Rozenberg, G., Salomaa, A. (eds.) Workshop on Membrane Computing. Lecture Notes in Computer Science, vol. 5391, pp. 187–203. Springer (2008)
8. García-Quismondo, M., Gutiérrez-Escudero, R., Pérez-Hurtado, I., Pérez-Jiménez, M.J., Riscos-Núñez, A.: An overview of p-lingua 2.0. In: Păun, G., Pérez-Jiménez, M.J., Riscos-Núñez, A., Rozenberg, G., Salomaa, A. (eds.) Workshop on Membrane Computing. Lecture Notes in Computer Science, vol. 5957, pp. 264–288. Springer (2009)
9. Haydon, P.G.: Glia: listening and talking to the synapse. Nature Reviews Neuroscience 2(3), 185–193 (2001)
10. Ionescu, M., Păun, G., Yokomori, T.: Spiking neural P Systems. Fundam. Inform. 71(2-3), 279–308 (2006)
11. Macías-Ramos, L.F., Pérez-Hurtado, I., García-Quismondo, M., Valencia-Cabrera, L., Pérez-Jiménez, M.J., Riscos-Núñez, A.: A p-lingua based simulator for spiking neural P Systems. In: Gheorghe, M., Păun, G., Rozenberg, G., Salomaa, A., Verlan, S. (eds.) Int. Conf. on Membrane Computing. Lecture Notes in Computer Science, vol. 7184, pp. 257–281. Springer (2011)
12. Pan, L., Păun, G.: Spiking neural P Systems with anti-spikes. International Journal of Computers, Communications and Control IV, 273–282 (09 2009), http://www.journal.univagora.ro/?page=article_details&id=372
13. Pan, L., Păun, G., Pérez-Jiménez, M.J.: Spiking neural P Systems with neuron division and budding. Science China Information Sciences 54(8), 1596–1607 (2011)
14. Pan, L., Wang, J., Hoogeboom, H.J.: Asynchronous extended spiking neural P Systems with astrocytes. In: Proceedings of the 12th international conference on Membrane Computing. pp. 243–256. CMC'11, Springer-Verlag, Berlin, Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-28024-5_17
15. Păun, A., Păun, G.: Small universal spiking neural P Systems. Biosystems 90(1), 48–60 (2007)
16. Păun, G.: Computing with membranes (P Systems): An introduction. In: Current Trends in Theoretical Computer Science, pp. 845–866 (2001)
17. Păun, G.: Spiking neural P Systems with astrocyte-like control. J. UCS 13(11), 1707–1721 (2007)

18. Păun, G., Rozenberg, G., Salomaa, A.: The Oxford Handbook of Membrane Computing. Oxford University Press, Inc., New York, NY, USA (2010)
19. Research Group on Natural Computing, University of Seville: The p–lingua website. http://www.p-lingua.org
20. Wang, J., Hoogeboom, H.J., Pan, L., Păun, G., Pérez-Jiménez, M.J.: Spiking neural P Systems with weights. Neural Comput. 22(10), 2615–2646, http://dx.doi.org/10.1162/NECO_a_00022