Chapter 8

Asynchronous Extended Spiking Neural P Systems with Astrocytes

Abstract
A variant of spiking neural P systems was recently investigated by the authors [Pan, Wang, Hoogeboom: Spiking Neural P Systems with Astrocytes, submitted], where astrocytes have excitatory and inhibitory influence on synapses. In this work, we consider this new system in the non-synchronized (i.e., asynchronous) mode: in any step, when a neuron is enabled, it is not obligatorily fired, making a global clock dispensable. It is proved that asynchronous spiking neural P systems with astrocytes are universal (when using extended rules).

8.1 Introduction

Spiking neural P systems (SN P systems, for short) are a class of distributed and parallel computation models inspired by the way neurons communicate by means of electrical impulses of identical shape (called spikes). SN P systems were introduced in [16], and then investigated in a large number of papers. Readers can refer to the recent handbook [35] for general information in this area.

Briefly, an SN P system consists of a set of neurons placed as the nodes of a directed graph. The content of each neuron consists of a number of copies of a single object type, called the spike. The rules assigned to neurons allow a neuron to send information to all other neurons that are connected to it.

Classically, an SN P system works in a synchronized manner. A global clock is assumed, and in each time unit, for each neuron with applicable rules one of these rules is nondeterministically chosen, which are then applied on the tick of the clock, for all neurons at the same time. While globally parallel, the work of the system is sequential in each neuron as (at most) one rule is applied in each neuron. Output can be defined in the form of the spike train produced by a specified output neuron, then measuring the timing between events, or counting the number of spikes in that neuron.

However, both from a mathematical point of view and from a neuro-biological point of view, it is rather natural to consider non-synchronized systems, where the application of rules is not obligatory, and no global clock is assumed. The neuron may remain unfired, maybe receiving spikes from the neighbouring neurons. Thus the unused rule may be used later, or it may loose its applicability. If further spikes made the rule non-applicable, then the computation continues in the new circumstances (maybe other rules are enabled now). With this motivation, asynchronous SN P systems were introduced in [4], and it was proved that asynchronous SN P systems with extended rules are equivalent with Turing machines.

Recently, a variant of spiking neural P systems with astrocytes was considered in [28], where an astrocyte can sense at the same time the spike traffic along several neighboring synapses. In an SN P system with astrocytes, each astrocyte ast has a given threshold t. Suppose that there are k spikes passing along the neighboring synapses. If k is larger than the threshold t, then the astrocyte ast has an inhibitory influence on the neighboring synapses, and the k spikes are suppressed (that is, they are removed from the system). If k is less than the threshold t, then the astrocyte ast has an excitatory influence on the neighboring synapses, the k spikes survive and pass to the destination neurons. If k equals t, then the astrocyte ast non-deterministically chooses an inhibitory or excitatory influence on the neighboring synapses.

Actually, astrocytes were already introduced into spiking neural P systems in [3, 31]. The functioning of astrocytes defined in [3, 28, 31] are different, although all of them are inspired by living astrocytes. In [3], two kinds of astrocytes are defined: excitatory astrocytes and inhibitory astrocytes. Astrocytes defined in [31] are a particular case of those considered in [3], with an inhibitory role; furthermore, the use of astrocytes defined in [31] adds a new degree of non-determinism to the functioning of the system, by the branching due to the non-deterministic choice of the surviving spike.

In this work, building [4], on we prove that spiking neural P systems with astrocytes are equivalent with Turing machines in the non-synchronized case. In our proof of universality, forgetting rules are not used, and they are actually replaced by the inhibitory influence of astrocytes. Moreover, all neurons in our construction work in a deterministic way (the nondeterminism in choosing the rules of a neuron is compensated by the nondeterminism from astrocytes).

In the universality proof in [4], the SUB modules of systems contain a parameter T, where T depends on the maximum number of SUB instructions that act on a same counter in the simulated register machine. However, in the present work, all modules of systems are constructed in a uniform way in the sense that all modules are independent of the particular register machine that is simulated (more specifically, for different register machines the combination of modules used is different; while the modules used are same).

The rest of this paper is organized as follows. In the next section, we introduce some necessary prerequisites. In Section 8.3, we present spiking neural P systems with astrocytes. In Section 8.4, we prove that spiking neural P systems with astrocytes are universal in non-synchronized case. Final remarks are presented in Section 8.5.

8.2 Prerequisites

Readers can refer to [38] for basic language and automata theory, as well as to [30] for basic membrane computing. We here only fix some necessary notions and notations.

For an alphabet V, V^* denotes the set of all finite strings over V, with the empty string denoted by λ. The set of all nonempty strings over V is denoted by V^+. When $V = \{a\}$ is a singleton, then we write simply a^* and a^+ instead of $\{a\}^*$, $\{a\}^+$.

Regular expressions are built starting from λ and single symbols using the operators union ($\cup$), concatenation ($\cdot$) and star ($*$). The language represented by expression E is denoted by $L(E)$, where $L(\lambda) = \varnothing$.

By NRE we denote the families of Turing computable sets of numbers. (Thus, NRE is the family of length sets of recursively enumerable languages.)

A register machine is a construct $M = (m, H, l_0, l_h, I)$, where m is the number of registers (each holding a natural number), H is the set of instruction labels, l_0 is the start label (labeling an ADD instruction), l_h is the halt label (assigned to instruction HALT), and I is the set of instructions. Each label from H labels exactly one instruction from I, thus precisely identifying it. The instructions are of the following form (and have the given intended meaning):

- $l_i : (\texttt{ADD}(r), l_j, l_k)$ (add 1 to register r and then nondertiministically go to one of the instructions with labels l_j, l_k),
- $l_i : (\texttt{SUB}(r), l_j, l_k)$ (if register r is non-zero, then subtract 1 from it, and go to the instruction with label l_j; otherwise, go to the instruction with label l_k),
- $l_h : \texttt{HALT}$ (the halt instruction, the computation stops).

A register machine M computes (generates) a number n in the following way. The register machine starts with all registers empty (i.e., storing the number zero). It applies the instruction with label l_0 and proceeds to apply instructions as indicated by labels (and, in the case of SUB instructions, by the contents of registers). If the register machine reaches the halt instruction, then the number n stored at that time in the first register is said to be computed by M. The set of all numbers computed by M is denoted by $N(M)$. It is known that register machines compute all sets of numbers which are Turing computable, hence they characterize NRE [27].

Without loss of generality, it can be assumed that l_0 labels an ADD instruction and that in the halting configuration all registers different from the first one are empty, and that the output register is never decremented during the computation (its content is only added to).

We use the following convention. When the power of two number generating/accepting devices D_1 and D_2 are compared, number zero is ignored; that is, we write $N(D_1) = N(D_2)$ when we actually mean that $N(D_1) - \{0\} = N(D_2) - \{0\}$ (this corresponds to the usual practice of ignoring the empty string in language and automata theory).

8.3 Spiking Neural P Systems with Astrocytes

In this section, we introduce the variant of asynchronous spiking neural P systems with astrocytes (ASNPA systems, for short). For more details on such kind of systems, please refer to [4] and [28].

An (extended) *spiking neural P system with astrocytes*, of degree $m \geq 1$, $\ell \geq 1$, is a construct of the form

$$\Pi = (O, \sigma_1, \dots, \sigma_m, syn, ast_1, \dots, ast_l, out), \text{ where:}$$

- $O = \{a\}$ is the singleton alphabet (a is called *spike*);
- $\sigma_1, \dots, \sigma_m$ are neurons, of the form $\sigma_i = (n_i, R_i), 1 \leq i \leq m$, where:
  a) $n_i \geq 0$ is the initial number of spikes contained in σ_i;
  b) R_i is a finite set of extended rules of the following form:
  $$E/a^c \to a^p$$
  where E is a regular expression over a, and $c \geq 1$, $p \geq 1$ with $c \geq p$. [1]
- $syn \subseteq \{1, 2, ..., m\} \times \{1, 2, ..., m\}$ with $(i, i) \notin syn$ for $1 \leq i \leq m$ (*synapses* between neurons);
- $ast_1, \dots, ast_l$ are *astrocytes*, of the form $ast_i = (syn_{ast_i}, t_i)$, where $1 \leq i \leq \ell$, $syn_{ast_i} \subseteq syn$ is the set of synapses controlled by the astrocyte ast_i, $t_i \in \mathbb{N}$ is the *threshold* of the astrocyte ast_i;
- $out \in \{1,2,...,m\}$ indicate the *output* neuron.

[1]We adhere to the practice where the number of spikes sent is not larger than the number of spikes removed from the neuron. This turns out to be only a technical constraint: a set of 'parallel' neurons is used to increase the number of spikes where necessary.

The rules $E/a^c \to a^p$ with $p \geq 1$ are called (extended) *firing* (we also say *spiking*) rules, and they are applied as follows. If the neuron σ_i contains k spikes, and $a^k \in L(E), k \geq c$, then rule $E/a^c \to a^p \in R_i$ can be applied. This means consuming (removing) c spikes (leaving $k - c$ spikes in neuron σ_i), the neuron is fired, sending p spikes out along all outgoing synapses. These spikes then reach the neighbouring neurons, unless they are intercepted by one of the astrocytes, as explained shortly below. If $L(E) = \{a^c\}$, then the rule is written in the simplified form $a^c \to a^p$.

In the non-synchronized case, considered here, the definition of a computational step in an ASNPA system is easy. In each moment, any neuron is free to use an applicable rule or not. Hence, if a rule in neuron σ_i is enabled at step t, it is not necessarily applied at that step, the neuron can remain still in spite of the fact that it contains rules which are enabled by its contents. If the enabled rule in neuron σ_i is not applied, and neuron σ_i receives new spikes, making the rule non-applicable, then the computation continues in the new circumstances (where maybe other rules are enabled).

Note that in "classical" membrane computing a global clock is assumed, marking the time for the whole system, and the functioning of the system is synchronized. In the synchronized mode, in each time unit, if a neuron σ_i is enabled, then one of its rules must be used. It is possible that two or more rules in a neuron are applicable at the same time, and in that case, only one of them is chosen non-deterministically. Thus the neurons together work in parallel (synchronously), while each separate neuron sequentially processes its spikes, using only one rule in each time unit.

Astrocytes work as follows. An astrocyte can sense the spike traffic along the neighboring synapses. For an astrocyte ast_i, suppose that there are k spikes passing along the neighboring synapses in syn_{ast_i}. If $k > t_i$, then the astrocyte ast_i has an inhibitory influence on the neighbouring synapses, and the k spikes are suppressed (that is, the spikes are removed from the system). If $k < t_i$, then the astrocyte ast_i has an excitatory influence on the neighboring synapses, all spikes survive and pass to the destination neurons. If $k = t_i$, then the astrocyte ast_i non-deterministically chooses an inhibitory or excitatory influence on the neighboring synapses.

It is important to point out that when a neuron spikes, its spikes immediately leave the neuron along all synapses. This in particular means that (i) if the spikes are not suppressed by any astrocyte, these spikes reach the target neurons simultaneously (as in the synchronized system, no time is needed for passing along a synapse from one neuron to another neuron); (ii) if any astrocyte has inhibitory influence on the synapse, then these spikes are suppressed simultaneously, which means that astrocytes remove the spikes in the synchronized way.

It is possible that two or more astrocytes control the same synapse. In this case, if all these astrocytes have excitatory influence on the synapse, then the spikes along this synapse can survive and pass to the destination neurons; if one of these astrocytes has inhibitory influence on the synapse, then the spikes along this synapse are suppressed and removed from the system.

A configuration of the system is described by the number of spikes present in each neuron. The initial configuration is defined by the number of initial spikes $n_1, \ldots, n_m$. Using the rules as described above, one can define transitions among configurations. Any sequence of transitions starting from the initial configuration is called a *computation*. A computation halts when it reaches a configuration where no rule can be used.

Because in asynchronous SN P systems with astrocytes, an enabled rule can be applied at any moment, the result of a computation can no longer be reliably defined in terms of the steps between two consecutive spikes as in one of the standard SN P system definitions. Therefore, in this work, the result of a computation is defined as the total number of spikes sent into the environment by the output neuron. Specifically, if there is a halting computation of an SN P system where the output neuron sends out exactly n spikes, then the system generates the number n.

Halting computations that send no spike out can be considered as generating number zero, but, in this work, we adopt the convention to ignore number zero when the computation power of two devices is compared. The reason is technical: we want to consider blocked computations without output as failure.

Note that the definitions in this paper specify systems with extended rules (introduced by [6], meaning that more than one spike can be emitted) but without forgetting rules (meaning that at least one spike must be emitted when firing). Astrocytes are an effective alternative method for removing spikes, so forgetting rules are not necessary in our considerations.

We denote by $N(\Pi)$ the set of numbers generated in the asynchronous way by an ASNPA system Π. By $N_{gen}^{asyn}SNPA(extend)$ we denote the family of such sets of numbers generated by ASNPA systems, using extended rules.

8.4 Universality of ASNPA Systems

In this section, ASNPA systems are proved to be universal in the generative mode.

Theorem 1
$NRE = N_{gen}^{asyn}SNPA(extend)$.

Proof
We show that $NRE \subseteq N_{gen}^{asyn}SNPA(extend)$; the converse inclusion is straightforward (simulating the system by a Turing machine). To this aim, we use the characterization of NRE by means of register machines used in the generative mode. Let us consider a register machine $M = (m, H, l_0, l_h, I)$. As introduced in Section 8.2, without any loss of generality, we may assume that in the halting configuration, all registers different from the output register 1 are empty, and that this register is never decremented during a computation. In what follows, a specific ASNPA system Π will be constructed to simulate the register machine M.

Each register r of M will be implemented by a neuron σ_r in Π, and if the register contains the number n, then the associated neuron will have $6n+6$ spikes. A neuron σ_{l_i} is associated with each label $l_i \in H$. We construct modules ADD and SUB to simulate the instructions of M, connecting the appropriate label-neurons, and introducing further auxiliary neurons $\sigma_{l_i^{(j)}}$, $j = 1, 2, 3, \ldots$. Finally an output module FIN is constructed to output computation results.

The modules will be specified in a graphical form. Neurons are denoted by rounded rectangles with the number of initial spikes inside; arrows between these rounded rectangles represent the synapses. An astrocyte is denoted by a rhombic box with "arms" touching the synapses; each arm indicates that the astrocyte controls the spike traffic of the corresponding touched synapse; the equation $t = k$ inside the rhombic box denotes that the astrocyte has threshold k.

In the initial configuration, all neurons are empty except that neuron σ_{l_0} associated with label l_0 of M has two spikes inside to set the system working, and each counter neuron σ_r contains 6 spikes representing empty counters. In general, when a label-neuron σ_{l_i}, $l_i \in H$, has two spikes inside, then it becomes active and the module associated with instruction l_i starts to work, simulating the instruction. Simulation is transferred to another instruction when the next label-neuron obtains two spikes.

Module ADD – simulating an ADD instruction $l_i : (\mathtt{ADD}(r), l_j, l_k)$.
Module ADD, shown in Figure 8.1, is composed of 2 astrocytes and 12 neurons: neuron σ_r for register r; neurons $\sigma_{l_i}, \sigma_{l_j}, \sigma_{l_k}$ for labels l_i, l_j, l_k; and 8 auxiliary neurons $\sigma_{l_i^{(1)}}, \ldots, \sigma_{l_i^{(8)}}$. The astrocyte $ast_{l_i}^{(1)}$ touches two synapses $(\sigma_{l_i^{(5)}}, \sigma_{l_i^{(7)}})$ and $(\sigma_{l_i^{(5)}}, \sigma_{l_i^{(8)}})$, and its threshold is 6. The astrocyte $ast_{l_i}^{(2)}$ touches synapse $(\sigma_{l_i^{(8)}}, \sigma_{l_k})$ and its threshold is 3.

Let us assume that at a computational step instruction $l_i : (\mathtt{ADD}(r), l_j, l_k)$ has to be simulated, with two spikes present in neuron σ_{l_i} and no spike in any other neurons, except in those neurons associated with registers.

Having two spikes inside, neuron σ_{l_i} may fire, sending 2 spikes out. These spikes will simultaneously go to neurons $\sigma_{l_i^{(1)}}$, $\sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$. Each of these three neurons can fire, and indeed must fire before neuron $\sigma_{l_i^{(4)}}$ is enabled. Then, with 6 spikes inside, neuron $\sigma_{l_i^{(4)}}$ sends 6 spikes to neurons σ_r (simulating the increase of the value of register r with 1) and $\sigma_{l_i^{(5)}}$. With 6 spikes inside, neuron $\sigma_{l_i^{(5)}}$ can use its rule $a^6 \to a^3$, sending 3 spikes to each of the neurons $\sigma_{l_i^{(6)}}$, $\sigma_{l_i^{(7)}}$ and $\sigma_{l_i^{(8)}}$.

At that moment there are 6 spikes passing along the synapses controlled by astrocyte $ast_{l_i^{(1)}}$ which has threshold 6. The astrocyte $ast_{l_i^{(1)}}$ non-deterministically chooses an inhibitory or excitatory influence on its controlled synapses. This choice determines whether the next label-neuron activated will eventually be σ_{l_j} or σ_{l_k}.

If astrocyte $ast_{l_i^{(1)}}$ chooses an inhibitory influence on its controlled synapses, then only neuron $\sigma_{l_i^{(6)}}$ receives 3 spikes from neuron $\sigma_{l_i^{(5)}}$. In turn it will send 2 spikes to neuron $\sigma_{l_i^{(8)}}$, which then can send its 2 spikes to neuron σ_{l_k} at one

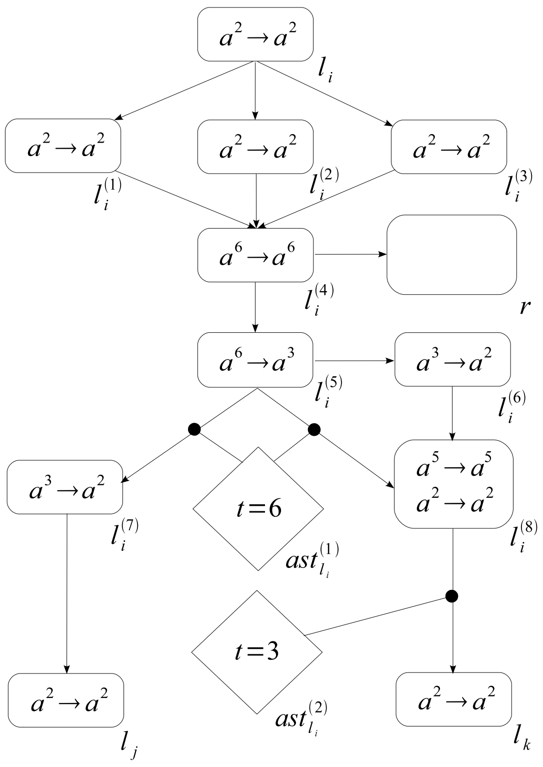

Figure 8.1: Module ADD simulating $l_i : (\texttt{ADD}(r), l_j, l_k)$

moment. The synapse $(\sigma_{l_i^{(8)}}, \sigma_{l_k})$ is controlled by astrocyte $ast_{l_i^{(2)}}$ but the number of spikes is less than the threshold 3 and the 2 spikes reach neuron σ_{l_k}. At that moment system Π starts to simulate the instruction l_k of M.

If astrocyte $ast_{l_i^{(1)}}$ chooses an excitatory influence on the synapses it controls, then 3 spikes reach each of the neurons $\sigma_{l_i^{(6)}}$, $\sigma_{l_i^{(7)}}$ and $\sigma_{l_i^{(8)}}$. With 3 spikes inside, neuron $\sigma_{l_i^{(7)}}$ can send 2 spikes to neuron σ_{l_j} starting the simulation of the instruction with label l_j of M. With 3 spikes inside, neuron $\sigma_{l_i^{(8)}}$ cannot apply any rule until further 2 spikes coming from neuron $\sigma_{l_i^{(6)}}$ arrive in neuron $\sigma_{l_i^{(8)}}$. Then, having 5 spikes inside, neuron $\sigma_{l_i^{(8)}}$ can send 5 spikes to neuron $\sigma_{l_i^k}$ at one moment. However, these 5 spikes passing along the synapse $(\sigma_{l_i^{(8)}}, \sigma_{l_i^k})$ are suppressed by astrocyte $ast_{l_i^{(2)}}$ and vanish from the system.

Note however that the system is asynchronous. There is no way to force the module to fire any of the neurons $\sigma_{l_i^{(6)}}$ and $\sigma_{l_i^{(8)}}$. Thus the possibility exists that 3 new spikes arrive (the next time that instruction l_i' is simulated) to neuron $\sigma_{l_i^{(6)}}$ and/or neuron $\sigma_{l_i^{(8)}}$ while that neuron is not yet empty. A careful case analysis now shows that this leads to more than 3 or 5 spikes in neuron $\sigma_{l_i^{(6)}}$ or neuron $\sigma_{l_i^{(8)}}$, respectively. This effectively means that label l_k becomes unreachable, blocking computations taking that branch. Most importantly however, it does not introduce halting but unwanted computations.

Therefore, from firing neuron σ_{l_i}, the system may non-deterministically fire one of neurons σ_{l_j} and σ_{l_k}, which correctly simulates the ADD instruction l_i : $(\mathtt{ADD}(r), l_j, l_k)$. Any other computation will stop without generating an output.

Module SUB – simulating a SUB instruction $l_i : (\mathtt{SUB}(r), l_j, l_k)$.
Module SUB, shown in Figure 8.2, is composed of 3 astrocytes $ast_{l_i}^{(1)}$, $ast_{l_i}^{(2)}$, $ast_{l_i}^{(3)}$ and 18 neurons: neurons σ_r for register r, neuron $\sigma_{l_i}, \sigma_{l_j}, \sigma_{l_k}$ for labels l_i, l_j, l_k, and 14 auxiliary neurons $\sigma_{l_i^{(1)}}, \sigma_{l_i^{(2)}}, \dots, \sigma_{l_i^{(14)}}$.

An instruction l_i is simulated in Π in the following way. Initially, neuron l_i has two spikes, and the other neurons are empty, except neurons associated with registers. Assume that at one moment neuron σ_{l_i} fires and sends 2 spikes to neurons $\sigma_{l_i^{(1)}}$ and $\sigma_{l_i^{(2)}}$. Neuron $\sigma_{l_i^{(1)}}$ will produce one spike and send it to neuron σ_r. There are the following two cases, depending whether the counter r contains a positive value or not.

Positive counter. If neuron σ_r has $6n+6$ $(n > 0)$ spikes (corresponding to the fact that the number stored in register r is positive), then after receiving one spike from neuron $\sigma_{l_i^{(1)}}$, neuron σ_r has $6n+7$ spikes, and the enabled rule $a^7(a^6)^+/a^7 \to a^6$ will fire, sending 6 spikes to neurons $\sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$, respectively. For neuron $\sigma_{l_i^{(2)}}$, there are following two cases.

(a) Neuron $\sigma_{l_i^{(2)}}$ has not consumed the first 2 spikes (received from neuron σ_{l_i}) before getting the second 6 spikes from neuron σ_r. Then, with 8 spikes inside, neuron $\sigma_{l_i^{(2)}}$ will send 6 spikes to neuron $\sigma_{l_i^{(3)}}$. This number is larger than the

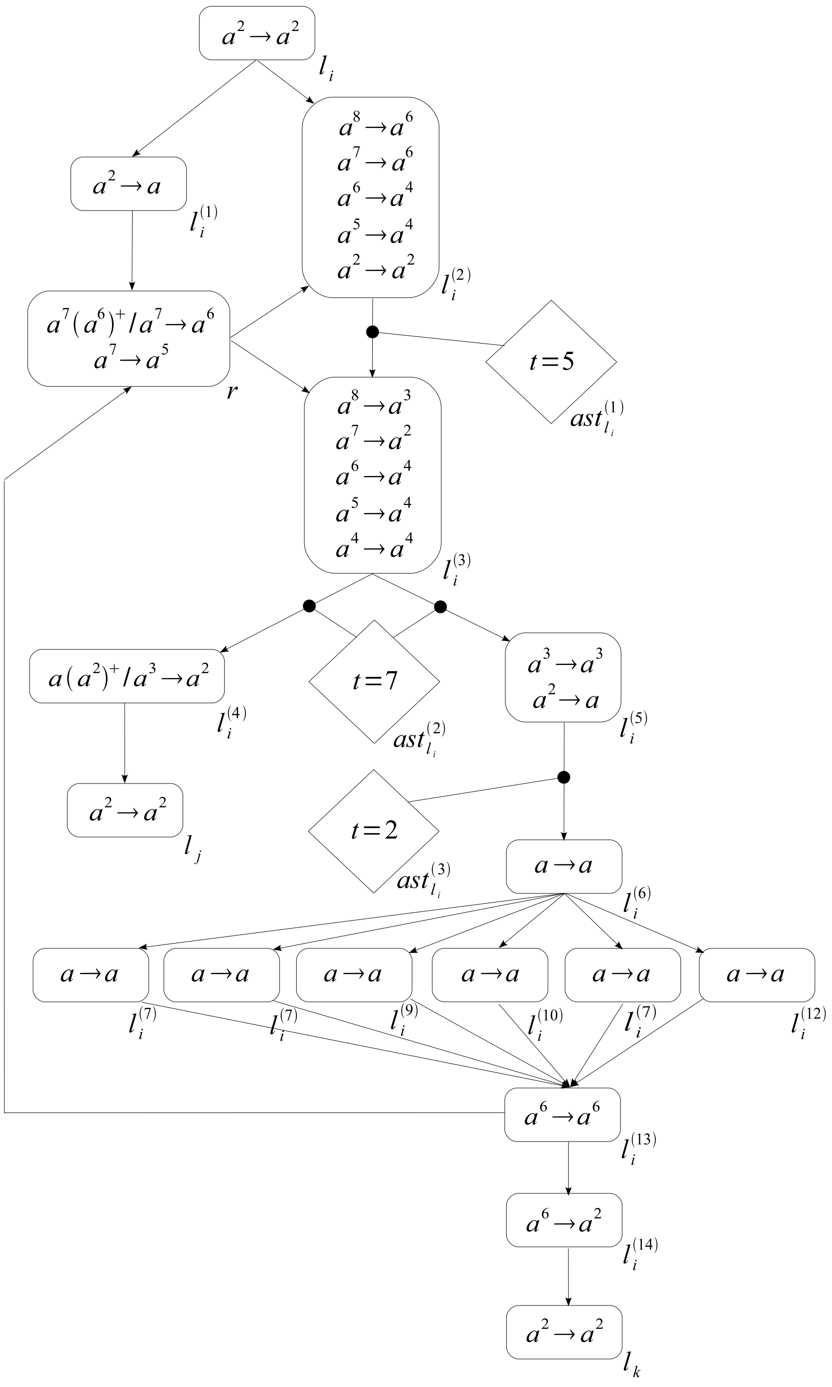

Figure 8.2: Module SUB simulating $l_i : (\texttt{SUB}(r), l_j, l_k)$

threshold 5 of astrocyte $ast_{l_i}^{(1)}$, thus the spikes are suppressed. With 6 spikes (from neuron σ_r) inside, neuron $\sigma_{l_i^{(3)}}$ sends 4 spikes to neurons $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(5)}}$, respectively. Again however, all the spikes sent out from neuron $\sigma_{l_i^{(3)}}$ are suppressed by astrocyte $ast_{l_i}^{(2)}$ as the total number of 8 spikes passing along the synapses is larger than the threshold 7 of astrocyte $ast_{l_i}^{(2)}$). In this case, neither of the neurons σ_{l_j} and σ_{l_k} receive a spike, which means that computations in system Π abort, outputting no spike into the environment.

(b) Neuron $\sigma_{l_i^{(2)}}$ receives the first 2 spikes from neuron σ_{l_i}, and rule $a^2 \to a^2$ is applied before the 6 spikes from neuron σ_r arrive in neuron $\sigma_{l_i^{(2)}}$, then later these 6 spikes will be consumed and 4 spikes will be sent to neuron $\sigma_{l_i^{(3)}}$. These numbers are less than the threshold and all spikes will arrive in neuron $\sigma_{l_i^{(3)}}$. When neuron $\sigma_{l_i^{(3)}}$ first receives the 2 spikes from neuron $\sigma_{l_i^{(2)}}$ it cannot apply any rule. When it receives the 6 spikes from neuron σ_r, the contents of $\sigma_{l_i^{(3)}}$ will be 8 spikes, while $\sigma_{l_i^{(2)}}$ contains 6 spikes. Now both $\sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$ are enabled. There are two situations:

- Enabled rule $a^8 \to a^3$ is not applied before the 4 spikes from neuron $\sigma_{l_i^{(2)}}$ arrive in neuron $\sigma_{l_i^{(3)}}$. Then neuron $\sigma_{l_i^{(3)}}$ accumulates 12 spikes, the number of spikes is greater than 8, thus, no rule is enabled and the computation aborts.
- Neuron $\sigma_{l_i^{(3)}}$ applies the rule $a^8 \to a^3$ before receiving the 4 spikes from neuron $\sigma_{l_i^{(2)}}$. Then 3 spikes are sent out to both neurons $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(5)}}$. So, there are 6 spikes passing along the synapses controlled by astrocyte $ast_{l_i}^{(2)}$. The number 6 is less than the threshold 7 of this astrocyte, hence these spikes can reach neurons $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(5)}}$. With an odd number of spikes inside (an even number plus the three just received), neuron $\sigma_{l_i^{(4)}}$ can fire by using the rule $a(a^2)^+/a^3 \to a^2$. Now neuron σ_{l_j} receives 2 spikes, and the system Π starts to simulate instruction l_j of M. At the same time neuron $\sigma_{l_i^{(4)}}$ again has an even number of spikes. In the other branch, with 3 spikes inside, neuron $\sigma_{l_i^{(5)}}$ can send 3 spikes to neuron $\sigma_{l_i^{(6)}}$. However, these 3 spikes are suppressed by astrocyte $ast_{l_i}^{(3)}$ and do not reach their destination neuron $\sigma_{l_i^{(6)}}$. Note that, after sending 3 spikes out, neuron $\sigma_{l_i^{(3)}}$ will then receive 4 spikes from neuron $\sigma_{l_i^{(2)}}$ and produce 4 spikes, but these spikes are suppressed by astrocyte $ast_{l_i}^{(2)}$ and cannot reach neurons $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(5)}}$.

Zero counter. If neuron σ_r initially has 6 spikes (corresponding to an empty register r), then at one moment after receiving the additional spike from $\sigma_{l_i^{(1)}}$, neuron σ_r has 7 spikes inside, sending 5 spikes to $\sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$, respectively. Neurons

$\sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$ are designed in such a way that their behaviour in the case of an empty counter is completely analogous to the non-zero case described above, except that the crucial numbers 6 and 8 of spikes are to be replaced by 5 and 7, leading to the same way incorrect computations abort, while a correct simulation will fire two spikes from $\sigma_{l_i^{(4)}}$. We omit the cases where the computation aborts, and only consider the correct branch (corresponding to the second item from (b) above in the non-zero case).

(b) Consider that neuron $\sigma_{l_i^{(2)}}$ contains 5 spikes received from σ_r, which means that it already sent 2 spikes to neuron $\sigma_{l_i^{(3)}}$. With 7 spikes inside, neuron $\sigma_{l_i^{(3)}}$ can apply the rule $a^7 \to a^2$.

- Neuron $\sigma_{l_i^{(3)}}$ first consumes 7 spikes and sends 2 spikes to neurons $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(5)}}$, respectively, and later sends 4 spikes out by using the rule $a^4 \to a^4$, then each neuron $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(5)}}$ receives only 2 spikes from neuron $\sigma_{l_i^{(3)}}$ (the 4 spikes sent out from neuron $\sigma_{l_i}^{(3)}$ are suppressed by astrocyte $ast_{l_i}^{(2)}$). Neuron $\sigma_{l_i^{(4)}}$ gets 2 spikes and keeps an even number of spikes, thus, no rule can be used and neuron σ_{l_j} receive no spikes. With 2 spikes inside, neuron $\sigma_{l_i^{(5)}}$ will send one spike to neuron $\sigma_{l_i^{(6)}}$, which of course is not blocked by astrocyte $ast_{l_i}^{(3)}$. Neuron $\sigma_{l_i^{(6)}}$ will send a spike to each of the neurons $\sigma_{l_i^{(7)}}, \ldots, \sigma_{l_i^{(12)}}$. The six neurons will send their spike to neuron $\sigma_{l_i^{(13)}}$. With 6 spikes inside, neuron $\sigma_{l_i^{(13)}}$ can send 6 spikes to neuron σ_r (reestablishing the number 0 in register r) and $\sigma_{l_i^{(14)}}$. Neuron $\sigma_{l_i^{(14)}}$ will send 2 spikes to neuron σ_{l_k}. This neuron becomes active, and the system Π can start to simulate instruction l_k of M.

The simulation of SUB instruction is correct: system Π starts from σ_{l_i} and ends in σ_{l_j} (if the number stored in register r is great than 0 and decreased by one), or in σ_{l_k} (if the number stored in register r is 0).

Note that there is no interference between the ADD modules and the SUB modules, other than correctly firing the neurons σ_{l_j} or σ_{l_k}, which may label instructions of the other kind. However, it is possible to have interference between two SUB modules. Specifically, if there are several SUB instructions l_t that act on register r, then when instruction $l_i : (\mathtt{SUB}(r), l_j, l_k)$ is simulated, neurons $\sigma_{l_t^{(2)}}$ and $\sigma_{l_t^{(3)}}$ not involved in the active instruction that is simulated nevertheless receive 6 or 5 spikes from neuron σ_r. We show that this does not lead to unwanted halting computations. In all cases, if the simulation does not continue as desired, a module will be blocked, which then forces the computation to abort when that module later might be started. Consider the number of spikes contained in each neurons $\sigma_{l_t^{(2)}}$ and $\sigma_{l_t^{(3)}}$ before receiving these spikes from neurons σ_r, we distinguish the following five cases.

(1) If both neurons $\sigma_{l_t^{(2)}}$ and $\sigma_{l_t^{(3)}}$ are empty before receiving 6 or 5 spikes from neuron σ_r, then after receiving these spikes, both of them contain 6 or 5 spikes (as we will see below, this is the initial state of case (2)). With 6 or 5 spikes inside, neuron $\sigma_{l_t^{(2)}}$ sends 4 spikes to neuron $\sigma_{l_t^{(3)}}$ and these spikes can reach their destination neuron $\sigma_{l_t^{(3)}}$. For neuron $\sigma_{l_t^{(3)}}$, there are two possible situations:

   (a) If neuron $\sigma_{l_t^{(3)}}$ consumes the first received 6 or 5 spikes from neuron σ_r before the 4 spikes from neuron $\sigma_{l_t^{(2)}}$ arrive, then it will send 4 spikes out and contains 4 spikes (this is the initial state of case (3)). With 4 spikes inside, neuron $\sigma_{l_t^{(3)}}$ can send 4 spikes out by using the rule $a^4 \to a^4$. However, all the spikes sent out from neuron $\sigma_{l_t^{(3)}}$ are suppressed by astrocyte $ast_{l_t^{(2)}}$ and none can reach their destination neurons $\sigma_{l_t^{(4)}}$ and $\sigma_{l_t^{(5)}}$.

   (b) If neuron $\sigma_{l_t^{(3)}}$ does not consume the first received 6 or 5 spikes before the 4 spikes arrive, then with 10 or 9 spikes inside (this is the initial state of case (4)), neuron $\sigma_{l_t^{(3)}}$ cannot apply any rule.

(2) If both neurons $\sigma_{l_t^{(2)}}$ and $\sigma_{l_t^{(3)}}$ have 6 or 5 spikes inside before receiving 6 or 5 spikes from neuron σ_r, then after receiving these spikes, both these neurons contain 12, 11 or 10 spikes, and no rule can be used.

(3) If neuron $\sigma_{l_t^{(2)}}$ has 0 spikes and neuron $\sigma_{l_t^{(3)}}$ has 4 spikes inside before receiving 6 or 5 spikes from neuron σ_r, then after receiving these spikes, neuron $\sigma_{l_t^{(2)}}$ can send 4 spikes to neuron $\sigma_{l_t^{(3)}}$. With 10 or 9 spikes inside, neuron $\sigma_{l_t^{(3)}}$ cannot fire any rule even it receives 4 spikes from neuron $\sigma_{l_t^{(2)}}$.

(4) If neuron $\sigma_{l_t^{(2)}}$ has 0 spikes and neuron $\sigma_{l_t^{(3)}}$ has 10 or 9 spikes inside before receiving 6 or 5 spikes from neuron σ_r, then similar to case (3), neuron $\sigma_{l_t^{(2)}}$ sends 4 spikes to neuron $\sigma_{l_t^{(3)}}$ and neuron $\sigma_{l_t^{(3)}}$ cannot apply any rule.

(5) If neuron $\sigma_{l_t^{(2)}}$ has 6 or 5 spikes and neuron $\sigma_{l_t^{(3)}}$ has 0 spikes inside (that is, after simulating the SUB instruction $l_t : (\texttt{SUB}(r), l_j, l_k)$, it is possible that there are 6 or 5 spikes left in neuron $\sigma_{l_t^{(2)}}$) before receiving 6 or 5 spikes from neuron σ_r, then after receiving these spikes, neuron $\sigma_{l_t^{(2)}}$ contains 12, 11 or 10 spikes and cannot apply any rule. Neuron $\sigma_{l_t^{(3)}}$ can send 4 spikes out but all the spikes are suppressed by astrocyte $ast_{l_t^{(2)}}$.

Therefore, the only computations in Π which can reach the neuron σ_{l_h} associated with the halting instruction of M are the computations which correctly simulate the instructions of M and correspond to halting computations in M.

Module FIN – outputting the result of a computation.

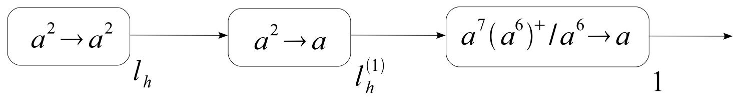

Figure 8.3: Module FIN (outputting the result of the computation)

Module FIN is shown in Figure 8.3. Assume that the computation in M halts, which means that the halting instruction is reached, and the number stored in register 1 is n (we may assume that $n \geq 1$, because number 0 is ignored when the computation power of computing devices is compared). This means that neuron σ_{l_h} in Π has two spikes inside and rule $a^2 \to a^2$ is enabled; neuron σ_1 has $6n+6$ spikes. When neuron σ_{l_h} fires it sends 2 spikes to neuron $\sigma_{l_h^{(1)}}$ which then (eventually) sends one spike to neuron σ_1, corresponding to the first register of M. From now on, neuron σ_1 can fire, sending out one spike for each 6 spikes present in it. If the number of spikes in neuron σ_1 is 7, then no rule is enabled in neuron σ_1. The computation in Π ends, and the number of spikes sent into the environment by system Π is n, which is exactly the number stored in register 1 of M when the computation of M halts.

From the description of the modules and their work, it is clear that the register machine M is correctly simulated by system Π. Therefore, $N(\Pi) = N(M)$. This completes the proof. ■

8.5 Conclusions and Remarks

In this work, we have considered spiking neural P systems with astrocytes used in a non-synchronized way: if a rule is enabled at some step, this rule is not obligatorily immediately used. Further spikes may make the rule non-applicable. Such systems are comparable to Petri nets which are also used in an asynchronous way. Without further features Petri nets are not computationally complete. One such feature is the use of inhibitor arcs in the net which indicate that a transition may fire only if the associated place is empty. We have proved that asynchronous SN P systems with astrocytes are universal. This can be compared to the case of Petri nets by observing that the astrocyte has an inhibiting kind of action.

There remain several open problems about asynchronous SN P systems with astrocytes. We just list two of them. We have used extended rules and do not know whether or not asynchronous SN P systems with astrocytes are Turing complete when using only standard rules. Second, the nondeterminism in our systems originates both from the uncertainty of the moment of firing and the two possible actions when the number of spikes equals the threshold of the astrocyte (following our earlier paper [28]). It would be natural to concentrate the nondeterminism in a single feature, i.e., to make the astrocytes purely deterministic.

Acknowledgements

We thank the three referees at CMC12 for their careful reading of the paper, and in particular for bringing relevant references to our attention.

The work of L. Pan and J. Wang was supported by National Natural Science Foundation of China (61033003 and 30870826), Ph.D. Programs Foundation of Ministry of Education of China (20100142110072), Fundamental Research Funds for the Central Universities (2010ZD001), and Natural Science Foundation of Hubei Province (2008CDB113 and 2008CDB180).