Neurons on wifi

David Orellana-Martín[1,2], Francis George C. Cabarle[1,2,3], Prithwineel Paul[4], XiangXiang Zeng[5], Rudolf Freund[6]

[1]Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
[2]SCORE lab, I3US, Universidad de Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
E-mail: dorellana@us.es, fcabarle@us.es
[3]Department of Computer Science, University of the Philippines Diliman,
1101 Quezon City, Philippines
E-mail: fccabarle@up.edu.ph
[4]Department of Computer Science and Engineering, Institute of Engineering and Management, University of Engineering and Management, New Town Rd., Kolkata, 700091, India
E-mail: prithwineel.paul@iem.edu.in
[5]Deparment of Computer Science, Hunan University, Changsha, China
E-mail: xzeng@hnu.edu.cn
[6]Faculty of Informatics, TU Wien, Favoritenstraße 9-11, 1040 Wien, Austria
E-mail: rudi@emcc.at

Summary. Spiking neural P systems, SN P systems in short, are membrane systems based on the third generation of neuron models (spiking neurons). Recent results in neuroscience highlight the importance of *extrasynaptic* activities of neurons, that is, features and functioning of neurons apart from their synapses. Previously it was thought that signals such as *neuropeptides* only assist neurons but such signals are given further importance more recently. Inspired by such recent results, we introduce the idea of *wireless SN P systems*, or *WSN P systems* in short. In WSN P systems no synapses exist, and we associate regular expressions for each neuron to decide which spikes it receives. We provide two semantics of how to "interpret" the spikes released by neurons. A specific register machine is simulated to show how different the programming style is with WSN P systems compared to SN P systems and other variants. The programming style emphasises a trade-off: WSN P systems can be more "flexible" in the sense that neurons are not limited by their synapses as before for sending spikes; the loss of the useful and directed graph, however, requires careful design of the rules and the regular expression associated with each neuron. For instance, in the present work we make use of prime numbers to create the expressions and rules of the neurons.

Keywords: Membrane computing, spiking neural P systems, extrasynaptic signalling, neuropeptides

1 Introduction

The present work introduces a variant of spiking neural P systems, in short SN P systems. SN P systems introduced in [1] are inspired by spiking neurons and their network: the processors are neurons which are the nodes in a directed graph; the edges are called synapses, which allow the communication between neurons of a single object a referred to as a spike; the neurons are spike processors which consume and produce spikes.

Some recent survey papers of SN P systems and variants include [2, 3] and more recently in [4]. Since their introduction, it is known that SN P systems are Turing complete. SN P systems can also solve **NP**-complete problems, trading time for space [5]. In the past decade or so many variants of SN P systems have been introduced depending on specific ingredients or features, mostly from biology. For instance the introduction of autapses [6], synaptic plasticity [7], synaptic schedules [8], neurogenesis [9].

Besides theoretical works, simulators of SN P systems and variants are used to support research or pedagogy, such as interactive and visual software in [10, 11] with the main page in [12], and recent tutorial in [13]. Solutions to hard problems are also implemented in parallel hardware such as in [14] which implements ideas from [15], with recent and some state-of-the-art results in [16].

In the present work we introduce the idea of wireless SN P systems, or WSN P systems in short. One general reference for the bio-inspiration of WSN P systems is from [17] with recent and detailed results from [18] and [19]. Briefly, such recent results emphasise the crucial and important role of neuronal activities outside of their synapses, hence their *wireless* features and functions. Such recent works focus their attention on a specific animal known as *C. elegans*.

The worm *C. elegans* is a model organism, that is, much is known about its biology including its nervous system due to its "simplicity" of several hundred neurons only. Despite the small size of this worm, its nervous system has interesting biochemical complexity with structural features shared by larger animals [19]. Due to better techniques and technology, more recently there are improved works to show how a *wireless network* (that is, without synaptic wiring) among nerve cells or neurons is able to operate [18, 19]. These recent works challenge the idea neurons communicate only or mainly through anatomical connections, that is, through their synapses [17]. Such recent works reveal new details of a *connectome* or wiring diagram among neurons, the *neuropeptidergic connectome*: a connectome which is equally important and perhaps more diverse than the synaptic connectome.

Furthermore, these recent works identify *neuropeptides*, the chemical messages released by neurons, as the basis for such wireless network among neurons. Neurons in the *C. elegans* worms can release neuropeptides, or have receptors for such neuropeptides. The wireless network formed from these *pairs of releasing* and *receiving neurons* is dense and decentralised, compared to the less dense and more centralised network of synapses [19]. Such pairs are responsible for existence of the wireless network, which means that neuropeptides are not simply random chemicals floating between neurons. Neuropeptides affect the neural system over larger scales of time and space, unlike synaptic signals restricted only to both sides of the synapse [19]

Previously it was thought that neuropeptides only assisted in synaptic communication. However, these recent works indicate the ubiquitous, important, and *direct role to neuron activation* of neuropeptides and the corresponding wireless network [17]. Neuropeptides are conserved and ancient chemicals in brains of many organisms, including humans brains, suggesting the pioneering work with *C. elegans* can at least reveal useful structures or principles for brain function [18, 19]. For instance, a recent technique allows to detect neuropeptides, which can assist in better understanding of both wired and wirless networks of neurons including those for humans [20].

We use the recent results mentioned as inspirations for *extrasynaptic* functions of neurons, that is, functioning without or outside the usual synapses. The contribution of the present work is the introduction of wireless SN P systems. No synapses are present in the neurons, while still using rules to consume and produce spikes. For each neuron we associate a regular expression to decide what "forms" of spikes the neuron can receive. We introduce two semantics for WSN P systems, based on the interpretation of the spikes released at each step by the neurons: the spike package semantic considers the spikes as individual packages as released by each neuron; the spike total semantic considers the sum of spikes released by all neurons. We show how to programme a specific WSN P system through the simulation of a specific register machine. Such a simulation emphasises the rather different way to programme WSN P systems compared SN P systems and variants, due to the associated expression for each neuron and the lack of synapses. In this way we note that the directed graph structure of SN P systems and variants is a very useful feature. Some "flexibility" is gained in the sense that the neurons are not limited to sending spikes only to neurons where their synapses connect. However, losing the directed graph makes the programming of the system more "involved" in the sense that more effort can be required to design the rules of each neuron.

The present work is organised as follows: in the next Section 2 we provide in an intuitive way an example of a WSN P system Π_1. We examine the computations of Π_1 under two semantics, the spike package and spike total semantics. In Section 3 we show how a WSN P system can simulate a small and specific register machine to highlight the rather different way to programme such systems. Lastly, in Section 4 we provide some conclusions and directions for further work.

2 An example with two semantics

In this section we consider an example, the system Π_1 shown in Figure 1. We use Π_1 to elaborate two semantics about wireless SN P systems. Briefly, Π_1 has 3 neurons, each labelled with a pair (i, E_i) for $1 \leq i \leq 3$. Each neuron has an associated regular expression to check what number of spikes it can receive. For instance, neurons σ_1 and σ_2 have $E_1 = E_2 = a$ which means they only receive spikes of the form $a^1 = a$ fired from other neurons, including from σ_1 itself. We note that the rule set of σ_2 is empty, so later we see the number of spikes inside it either remain the same or increase.

We omit the definition, syntax, and semantics standard to SN P systems. The reader is referred instead for instance to the seminal paper [1], in open access tutorials or surveys as in [21, 2], or the dedicated chapter of the handbook in [22].

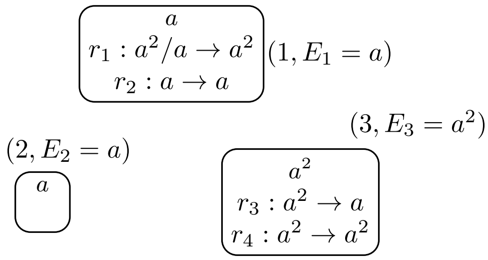

Fig. 1: Π_1 is an example of a wireless SN P system.

2.1 Semantic 1: spike package

The semantic 1 we first consider, which we refer to as *spike package semantic,* considers only in spikes arriving in "packages" sent by neurons in the environment. Consider two neurons which fire at the same step t: let neuron σ_i and σ_j have regular expressions $E_i = a^m$ and $E_j = a^n$ associated, respectively, for $n, m \geq 1$; neuron σ_i and σ_j fire a^n and a^m spikes at step t, respectively. At the next step $t + 1$, neuron σ_i receives the a^m spikes from neuron σ_j, and vice-versa. That is, while at step t there is a total of $n + m$ spikes in the environment due to the firing of both neurons: in spike package semantic we only consider packages or groups of the spikes and not the total spikes in the environment. We consider the spike total semantic as semantic 2 in Section 2.2 later.

Let us now apply the spike package semantic to the system Π_1 in Figure 1. To help with clarifying the computation of Π_1 we refer to the configuration tree in Figure 2 under spike package semantic.

The initial configuration of Π_1, according to the total ordering of 1, 2, and 3 of the neurons, is $C_0 = \langle 1, 1, 2 \rangle$. That is neurons 1, 2, and 3 each have 1, 1, and 2 spikes, respectively. Due to C_0 and the nondeterminism in Π_1 found only in neuron σ_3, there is a choice between applying rule r_2, and either r_3 or r_4.

If rule r_2 is applied one spike is consumed in neuron σ_1, and sent to both σ_1 and neuron σ_2 due to their associated regular expressions $E_1 = E_2 = a$. Applying r_3 means σ_3 consumes two spikes but produces only one spike. Again the single spike from σ_3 arrives at σ_1 and σ_2 due to their regular expressions. Hence, we have the transition $C_0 \overset{r_2 r_3}{\Longrightarrow} C_{1,0} = \langle 2, 3, 0 \rangle$, that is, by applying r_2 and r_3 we obtain configuration $C_{1,0}$ from C_0.

Consider now if we apply r_2 and r_4 instead. The effect applying of r_2 is still to return a spike to σ_1 and to increase the spikes in σ_2. The effect of r_4 is *reflexive*, that is, in neuron σ_3 two spikes are consumed and then returned to itself since $E_3 = a^2$. Hence, we have the transition $C_0 \overset{r_2 r_4}{\Longrightarrow} C_{1,1} = \langle 1, 2, 2 \rangle$, that is, by applying r_2 and r_4 we obtain configuration $C_{1,1}$ from C_0.

As seen in the configuration tree in Figure 2, each branch of computation of Π_1 is nonhalting, that is, Π_1 always arrives at a configuration where some rule is applied. The number of spikes in neuron σ_2 continue to increase. More precisely, we have transition $\langle 2, b, 0 \rangle \overset{r_1}{\Longrightarrow} \langle 1, b, 2 \rangle$, transition $\langle 1, b, 2 \rangle \overset{r_2 r_3}{\Longrightarrow} \langle 2, b + 2, 0 \rangle$, or transition $\langle 1, b, 2 \rangle \overset{r_2 r_3}{\Longrightarrow} \langle 1, b + 1, 2 \rangle$ for some $b \geq 1$.

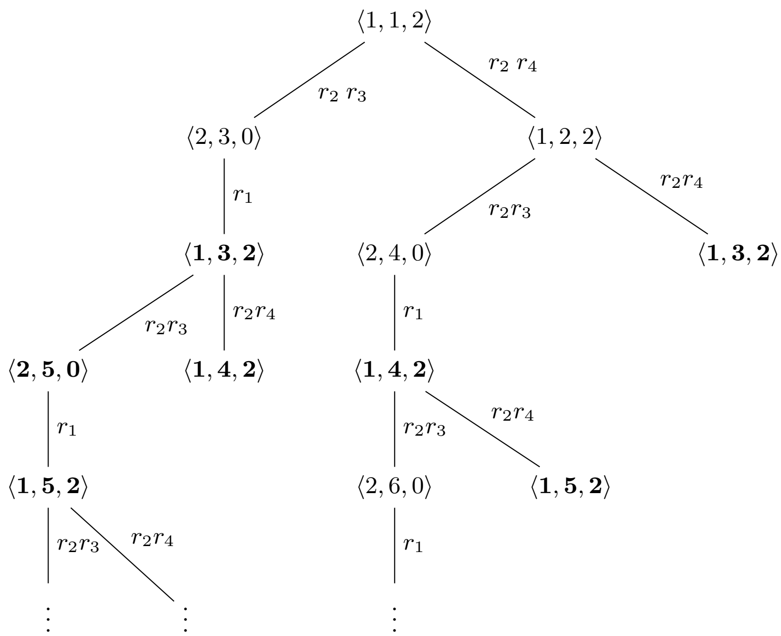

Fig. 2: A tree of configurations of Π_1 in Figure 1 using semantic 1 (spike package semantic). The initial configuration is $\langle 1, 1, 2\rangle$. Except for $\langle 1, 1, 2\rangle$, each *node* in the tree is a next configuration by applying the rules labelling the connecting *edge*. Nodes or configurations in bold are nodes repeated elsewhere in the portion of the tree shown.

2.2 Semantic 2: spike total

We continue the same notation at the start of Section 2.1 to consider the total spike semantic. Recall we have neurons with labels and their associated expressions as $\sigma_i = (i, E_i = a^m)$ and $\sigma_j = (j, E_j = a^n)$ for $n, m \geq 1$. At step t neurons σ_i and σ_j fire n and m spikes, respectively. Thus we have a total of $n + m$ spikes in the environment. In the next step $t + 1$, no neuron receives any spikes since $a^{n+m} \notin L(E_i)$ and $a^{n+m} \notin L(E_j)$. That is, none of the regular expressions of both neurons describe the total number of spikes in the environment.

Consider now the same SN P system Π_1 from Figure 1 but under the total spikes semantic. The configuration tree of Π_1 is now given by Figure 3. From the same initial configuration $C_0 = \langle 1, 1, 2\rangle$ the computation proceeds in a different way. The transition $C_0 \overset{r_2 r_4}{\Longrightarrow} C_{1,1} = \langle 0, 1, 0\rangle$ is a halting configuration, that is, no more rules can be applied in Π_1. Only the subtree with transition $C_0 \overset{r_2 r_3}{\Longrightarrow} C_{1,0} = \langle 0, 1, 2\rangle$ continues to infinitely grow the number of spikes in neuron σ_2. Actually after configuration $C_{2,0} = \langle 1, 2, 0\rangle$ only rule r_2 can be applied in a nonhalting computation.

We note that the effect of applying rules r_2 and r_4 from C_0 is to release a total of a^3 spikes in the environment followed by the halting of Π_1. Since we use the total spikes semantic no neuron receives these spikes in the next step because no neuron has an expression which includes a^3.

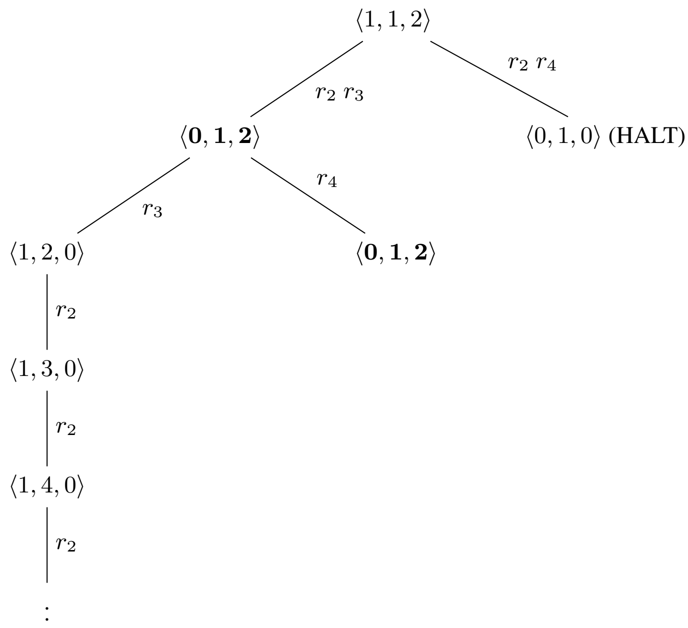

Fig. 3: Configuration tree for Π_1 in Figure 1 using semantic 2 (total spikes semantic). As in Figure 2, edges between nodes (configurations) are labelled by the rules applied from the source to destination nodes. Also, configurations in bold means they are repeated elsewhere in the tree.

3 Programming WSN P systems

Let us consider a small programme with some register machine M to give us an idea how to programme a WSN P system, including their similarities and differences with SN P systems and their other variants. It is known that register machines compute the set of all Turing computable sets of numbers [23]. We do not go into the details of register machines here, and refer the reader instead to [23] as well as to [1, 22] for the usual style of proofs with register machines. Consider the following instructions of a register machine M :

$l_1 : (SUB(r_1), l_2, l_3),$
$l_2 : (ADD(r_1), l_1, l_3),$
$l_3 : HALT.$

We simulate the instructions of M using a WSN P system Π_M with the following details. We map prime numbers to elements of M and use the mapping as *addresses* in Π_M. The idea of addresses and simulation is made clear in a moment. In general, for elements of any register machine we use the total order $l_1, l_2, \ldots, r_1, r_2, \ldots$. Specific to M we have the following mapping of its elements to prime numbers:

$$p_{l_1} = 7, p_{l_2} = 11, p_{l_3} = 13, p_{r_1} = 17.$$

That is, starting from instruction l_1 of M we map it to the prime number $p_{l_1} = 7$, followed by mapping l_2 and l_3 to $p_{l_2} = 11$ and $p_{l_3} = 13$, respectively. After mapping prime numbers to all instructions of M, we map the next prime numbers to registers: there is only one register in M mapped to $p_{r_1} = 17$.

In general, the mapping we use for the content of register $r_i = n$ is having $a^{2p_{r_i}n}$ spikes in the neuron σ_{r_i}. Following the mapping of prime numbers above to elements of M: if $r_1 = n$ the associated neuron σ_{r_1} has $a^{2p_{r_1}n} = a^{2(17)n}$ spikes.

3.1 Simulating a SUB instruction

Now we provide the SUB module of Π_M to simulate instruction l_1 of M. The SUB module consists of the following neurons and their contents. We note that the contents of a neuron $\sigma_i = (a^n, R_i, E_i)$ consists of its initial number of n spikes, its rule set R_i, and the associated regular expression E_i.

$\sigma_{l_1} = (a^7, R_{l_1}, E_{l_1} = a^7)$,
$\sigma_{aux_{1,1}} = (\lambda, R_{aux_{1,1}}, E_{aux_{1,1}} = a^{2(17)}(a^{17})^+)$,
$\sigma_{r_1} = (a^{2(17)n}, R_{r_1}, E_{r_1} = a^{17} \cup a^{2(17)})$.

The rule sets of each neuron we list as follows.

$R_{l_1} = \{a^7 \to a^{7(17)}\}$,
$R_{aux_{1,1}} = \{a^{7(17)}/a^{7(17-1)} \to a^{17}, a^{7+3(17)} \to a^{11}, a^{7+5(17)} \to a^{13}\}$,
$R_{r_1} = \{(a^{2(17)})^+ a^{17}/a^{3(17)} \to a^{3(17)}, a^{17} \to a^{5(17)}\}$.

The rules in each rule set are written in an explicit way with their superscripts, to make it easier to see the idea of the simulation. For instance, in simulating instruction l_1, neuron σ_{l_1} has only one rule releasing $a^{p_{l_1}(p_{r_1})} = a^{7(17)}$ spikes to mean the following: the source of spikes is σ_{l_1} with σ_{r_1} as destination. To simulate the next instruction, neuron $\sigma_{aux_{1,1}}$ releasing either $p_{l_2} = 11$ or $p_{l_3} = 13$ spikes means the destination neuron is either σ_{l_2} or σ_{l_3}, respectively.

Now we simulate instruction l_1 of M by the SUB module of Π_M as follows. Consider a total order of neurons in the SUB module of Π_M as $\sigma_{l_1}, \sigma_{aux_{1,1}}, \sigma_{r_1}$. From the above description, the initial configuration at time step $t = 0$ of the total order is given by $C_0 = \langle 7, 0, 2(17)n \rangle$.

At step $t = 1$, the a^7 spikes in neuron σ_{l_1} start the computation by applying the single rule in the neuron: all $p_{l_1} = 7$ spikes are consumed and $7(17) = p_{l_1}(p_{r_1})$ spikes are produced. The reason for 7(17) spikes is to indicate that instruction l_1 sends its spikes to perform subtraction operation on register r_1. At step 1 have the configuration $C_1 = \langle 0, 7(17), 2(17)n \rangle$. When the spikes have been sent, only the auxiliary neuron $\sigma_{aux_{1,1}}$ receives the spikes from σ_{l_1} since only the regular expression associated with $\sigma_{aux_{1,1}}$ makes a match. That is, we have $a^{7(17)} \in L(E_{aux_{1,1}})$.

At step $t = 2$, only the rule $a^{7(17)}/a^{7(17-1)} \to a^{17}$ of $\sigma_{aux_{1,1}}$ is applied: it consumes $7(17-1)$ spikes and produces 17 spikes is received only by neuron σ_{r_1}. At step 2 we have the configuration $C_2 = \langle 0, 7, 2(17)n + 17\rangle$.

At step $t = 3$ only neuron σ_{r_1} can apply a rule. Depending on the value of n in register r_1 of M, we have the following two cases:

1. if $n > 0$, this means that before $t = 2$, neuron σ_{r_1} has $2(p_{r_1})n = 2(17)n \geq 34$ spikes. Let $n = 1$. Then, receiving 17 spikes means the total spikes in σ_{r_1} at step $t = 2$ is $17 + 34n = 51$ spikes.
   The first rule of σ_{r_1} is applied since $a^{51} \in L((a^{2(17)})^+a^{17})$. Applying the rule consumes $3(17)$ spikes, no spikes remain in σ_{r_1} since $51 - 3(17) = 0$. In this way, as the number in register r_1 is reduced from n to $n-1$, the number of spikes in σ_{r_1} is reduced from $a^{2(17)n}$ to $a^{2(17)(n-1)}$. The rule also produces $a^{3(17)}$ spikes which in step $t = 4$ only $\sigma_{r_{1,1}}$ receives.
   At the moment $t = 4$ the configuration is $C_4 = \langle 0, 7+3(17), 2(17)(n-1)\rangle$ and only $\sigma_{r_{1,1}}$ can apply a rule: the neuron applies the rule $a^{7+3(17)} \to a^{11}$ to consume all of its spikes and to activate the next module to simulate instruction l_2 associated with $p_{l_2} = 11$.
2. if $n = 0$, before step $t = 2$ neuron σ_{r_1} has no spikes. Receiving 17 spikes means the total spikes in σ_{r_1} at moment $t = 3$ is 17 spikes: the neuron applies its rule $a^{17} \to a^{5(17)}$ to consume all spikes and send spikes only to $\sigma_{r_{1,1}}$. In this way, as the number in register r_1 is 0, the number of spikes in σ_{r_1} remains 0 also.
   At the moment $t = 4$ the configuration is now $C_4 = \langle 0, 7+5(17), 0\rangle$, with only $\sigma_{r_{1,1}}$ applying a rule: the rule $a^{7+5(17)} \to a^{13}$ is applied, consuming all spikes. At the next step the simulation of instruction l_3 associated with $p_{l_3} = 13$ begins.

Thus, the subtraction instruction l_1 of M is correctly simulated: if register r_1 contains a nonzero value it is decremented and the next instruction is l_2, otherwise r_1 remains zero and l_3 is the next instruction. We note that there is no interference in the case when there is more than one subtraction instruction associated with r_1. The mapping of prime numbers over a total ordering on M described above, and the "addresses" of each neuron based on the mapping allows no wrong simulation. Such addresses we use not only in the regular expressions associated with each neuron, but also in the spikes released by each neuron.

3.2 Simulating an ADD instruction

This section is devoted to the ADD module of Π_M to simulate instruction l_2 of M provided at the start of Section 3. The ADD module consists of the following neurons and their contents. For simulating this, we must include new rules in $R_{aux_{1,1}}$

$\sigma_{l_2} = (\lambda, R_{l_2}, E_{l_2} = a^{11})$,

The rule sets of each neuron we list as follows.

$R_{l_2} = \{a^{11} \to a^{11(17)}\}$,

$R_{aux_{1,1}} = R_{aux_{1,1}} \cup \{a^{11(17)}/a^{11(17-1)} \to a^{2(17)}, a^{11} \to a^7, a^{11} \to a^{13}\}$

Let us suppose that, at step $t = k$, a^{11} spikes arrive to neuron σ_{l_2}. Then, the simulation of the instruction l_2 starts. Consider a total order of neurons in the ADD module of Π_M as

$\sigma_{l_2}, \sigma_{aux_{1,1}}, \sigma_{r_1}$. From the above description, the configuration at the time $t = k$ neuron σ_{l_2} receives a^{11} spikes of the total order is given by $C_k = \langle 11, 0, 2(17)n \rangle$.

At step $t = k + 1$ the a^{11} spikes in neuron σ_{l_2} start the simulation by applying the single rule in the neuron: all $p_{l_2} = 11$ spikes are consumed and $11(17) = p_{l_1}(p_{r_1})$ spikes are produced. Similar to the SUB instruction, the reason for $11(17)$ spikes is to indicate that instruction l_2 sends its spikes to perform addition to register r_1. When the spikes have been sent, only the auxiliary neuron $\sigma_{aux_{1,1}}$ receives the spikes from σ_{l_1} since only the regular expression associated with $\sigma_{aux_{1,1}}$ makes a match. That is, we have $a^{11(17)} \in L(E_{aux_{1,1}})$.

At step $t = k + 2$, only the rule $a^{11} \rightarrow a^{11(17)}$ of $\sigma_{aux_{1,1}}$ is applied: it consumes $11(17-1)$ spikes and produces $2(17)$ spikes. Only neuron σ_{r_1} will receive the spikes. Thus, $C_{k+2} = \langle 0, 11, 2(17)(n+1) \rangle$.

At step $t = k + 3$, both rules $a^{11} \rightarrow a^7$ and $a^{11} \rightarrow a^{13}$ are applicable, so one of them is selected in a non-deterministic way. If the first one is applied, a^7 spikes are be fired, matching with the regular expression of neuron σ_{l_1}. Otherwise, rule a^{13} spikes are sent to neuron σ_{l_3} mapped to $p_{l_3} = 13$. In the first case, the SUB instruction is simulated again, while in the second case the output must be produced followed by the halting of Π_M.

Thus, the addition instruction l_2 of M is correctly simulated: the value of register r_1 is augmented by 1 and the next instruction is selected from the set $\{l_1, l_3\}$ in a non-deterministic way. No interference with rules from the SUB instruction is found. The regular expressions always match with the prime number p_{l_2} corresponding with instruction l_2. That is, p_{l_2} spikes are never released while simulating the SUB instruction.

Before we end the present section on programming Π_M to simulate M we make a few more notes. First we omit the explicit simulation of instruction l_3 to halt M. In Π_M, simulating a halt instruction requires the release of the output to the environment. In the above description of Π_M we assume the use of spike package semantic as in Section 2.1. It seems to be the case that Π_M as described above can still simulate M under the spike total semantic in Section 2.2.

4 Final remarks

We introduced yet another variant of SN P systems we refer to as wireless SN P systems, or WSN P systems in short. Several kinds of novelty can be found in WSN P systems. The further movement away from a fixed or static graph motivated especially by recent and exciting discoveries in neuroscience. That is, our increasing knowledge of extrasynaptic signalling, of neuropeptides and their important influence in neuronal activities. WSN P systems go against the traditional directed graph used in neural systems or networks, introducing two semantics how the "floating" spikes are received. We associate regular expressions to each neuron allowing neurons to distinguish which spikes to accept or reject. Both semantics, the spike partial and spike total, are bio-inspired. The semantics also bear some resemblance to packets of data among networks of computers that connect for instance wireless networks and the Internet.

The use of forgetting rules of the form $a^s \rightarrow \lambda$, are common to SN P systems and many variants. Forgetting rules are used to remove spikes without producing spikes, but such rules may not be necessary in WSN P systems. A way to avoid such rules is to use a rule $a^s \to a^x$ where x is not found in any regular expression of any neuron. In this way we still remove the s number of spikes, but more care needs to be given using the spike total semantic (Section 2.2). The output neuron mentioned at the end of Section 3.2 needs to be a distinguished neuron, in order to obtain the output of the system.

In many variants of SN P systems the delay feature is common: there can be a nonzero delay from releasing a spike and the spike arriving to another neuron. It is known, see e.g. [24], that the delay feature is not required for universality, but can be useful for instance in modelling [25]. It is interesting to see the role of delays in WSN P systems. Other common features of SN P systems and variants include the lack of reflexive synapses, and restricting the produced spikes of a neuron to be at most the consume spikes. A variant known as SN P systems with autapses allows reflexive synapses although this variant has a static and directed graph [6]. For restricting the produced spikes to be less or equal to the consumed spikes, perhaps this can be achieved by using more time in the computation, and more neurons to generate the required spikes.

Regarding the semantics in Section 2, it is interesting to see which types of problems or computations one semantics has an advantage over the other. As seen in the configuration trees in Figure 2 and Figure 3, for the same Π_1 the computations are rather different. Another interesting extension or semantic for WSN P systems is the idea of decay or "attenuation" of spikes: it is assumed that spikes (especially if delays are introduced) can "float" without change for an arbitrary duration in time or distance in space between neurons. It is interesting to introduce such decay or attenuation in WSN P systems, similar to decay of electromagnetic signals used in wireless networks of computers. Previously, decaying spikes were considered in SN P systems [26].

In programming Π_M we notice its operation is sequential, that is, at each step at most one neuron applies a rule. The sequential restriction or normal form has been applied to SN P systems as early as in [27], and more recently with variants having dynamic topologies in [28, 29]. It is interesting what kinds of restrictions and computations can(not) be obtained when more parallelism is involved in the system in terms of neurons, rules, etc.

Another interesting direction is to consider matrix representations of WSN P systems, as done with SN P systems in [30] and more recently in [31, 11]. Such representations allow for faster simulations, such as parallel processors [16, 32] web browsers [10, 11, 33], and their automatic design [34, 35]. The related variant with matrix representation seems to be SNPSP systems in [36]. SNPSP systems introduce plasticity to allow adding or removing of synapses, introduced in [7]. Another variant known as SNP systems with scheduled synapses (in short, SSNP systems) has synapse dynamism, by assigning schedules or (range of) time steps when synapses exist or not. Besides SNPSP systems and SSNP systems, another related variant are extended SN P systems in [37] which also have no fixed and directed graph structure. It is also interesting to consider WSN P systems in the formal framework of [38] for membrane systems and related models.

A few other lines of investigation on computing power to consider are the following. Computing languages with WSN P systems, for instance in [39, 40]. Providing "small" WSN P systems as in [41, 42]. Normal forms such as restricting the types of regular expressions, with optimal results in [24]. In the case of WSN P systems not only are there expressions in rules but also those associated with the neurons. Creating homogeneous systems as in [43, 44] is also worth investigating: the expressions associated for some neurons must be distinct but not for others; also the number of produced spikes may need to be heterogeneous for some rules, unlike previous works on homogeneous SN P systems where each neuron has the same rule set.

Besides computing power, computing efficiency is interesting to consider with WSN P systems. For instance how to solve **NP**-complete problems in a (non-)uniform way [45, 5]. An interesting extension is the feature to allow creation of new neurons as in [9, 46] or using the idea of pre-computed resources [47]. Real world applications can perhaps benefit from WSN P systems with neurons having the ability to "distinguish signals" using their associated expressions. Applications may include improvements on intrusion detection [48] and skeletonizing images [49]. More directions and open problems can be derived from [2, 4].

We end the present work by highlighting, based on the ideas here presented, that the directed graph structure of an SN P system seem to be powerful, at least useful, in programming the system. Losing such directed graph as shown in WSN P systems we need to use regular expressions for each neuron. Besides, here we use a mapping of prime numbers for simulating a register machine: a rather unconventional way of simulation at least in terms of the usual way of simulating register machines with such membrane systems. These ideas show that the programming of WSN P systems are quite different and interesting compared to SN P systems and their many variants.

Acknowledgements

F.G.C. Cabarle is supported by the QUAL21 008 USE project, "Plan Andaluz de Investigación, Desarrollo e Innovación" (PAIDI) 2020 and "Fondo Europeo de Desarrollo Regional" (FEDER) of the European Union, 2014-2020 funds. X. Zeng was supported by National Natural Science Foundation of China (Grant Nos. 62122025, U22A2037, 62250028)

References

1. Ionescu, M., Păun, G., Yokomori, T.: Spiking neural p systems. Fundamenta informaticae **71**(2, 3) (2006) 279–308
2. Leporati, A., Mauri, G., Zandron, C.: Spiking neural p systems: main ideas and results. Natural Computing **21**(4) (2022) 629–649
3. Fan, S., Paul, P., Wu, T., Rong, H., Zhang, G.: On applications of spiking neural p systems. Applied Sciences **10**(20) (2020) 7011
4. Cabarle, F.G.C.: Thinking about spiking neural p systems: some theories, tools, and research topics. Journal of Membrane Computing (2024) 1–20
5. Leporati, A., Mauri, G., Zandron, C., Păun, G., Pérez-Jiménez, M.J.: Uniform solutions to sat and subset sum by spiking neural p systems. Natural computing **8**(4) (2009) 681–702
6. Song, X., Valencia-Cabrera, L., Peng, H., Wang, J.: Spiking neural p systems with autapses. Information Sciences **570** (2021) 383–402

7. Cabarle, F.G.C., Adorna, H.N., Pérez-Jiménez, M.J., Song, T.: Spiking neural p systems with structural plasticity. Neural Computing and Applications **26** (2015) 1905–1917
8. Cabarle, F.G.C., Adorna, H.N., Jiang, M., Zeng, X.: Spiking neural p systems with scheduled synapses. IEEE Transactions on Nanobioscience **16**(8) (2017) 792–801
9. Wang, J., Hoogeboom, H.J., Pan, L.: Spiking neural p systems with neuron division. In: Membrane Computing: 11th International Conference, CMC 2010, Jena, Germany, August 24-27, 2010. Revised Selected Papers 11, Springer (2011) 361–376
10. Gulapa, M., Luzada, J.S., Cabarle, F.G.C., Adorna, H.N., Buño, K., Ko, D.: Websnapse reloaded: The next-generation spiking neural p system visual simulator using client-server architecture. In: Proceedings of the Workshop on Computation: Theory and Practice (WCTP 2023), Atlantis Press (2024) 434–461
11. Gallos, L., Sotto, J.L., Cabarle, F.G.C., Adorna, H.N.: Websnapse v3: Optimization of the web-based simulator of spiking neural p system using matrix representation, webassembly and other tools. In: Proceedings of the Workshop on Computation: Theory and Practice (WCTP 2023), Atlantis Press (2024) 415–433
12. : Websnapse page (2023) `https://aclab.dcs.upd.edu.ph/productions/software/websnapse`.
13. Ko, D., Cabarle, F.G.C., De L Cruz, R.T.: WebSnapse Tutorial: A Hands-On Approach for Web and Visual Simulations of Spiking Neural P Systems. In: Bulletin of the International Membrane Computing Society. Volume 16. (December 2023) 137–153
14. Hernández-Tello, J., Martínez-Del-Amor, M.A., Orellana-Martín, D., Cabarle, F.G.: Sparse matrix representation of spiking neural p systems on gpus. In: International Conference on Membrane Computing, Chengdu, China and Debrecen, Hungary (August 2021) 316–322
15. Martínez-Del-Amor, M.Á., Orellana-Martín, D., Pérez-Hurtado, I., Cabarle, F.G.C., Adorna, H.N.: Simulation of spiking neural p systems with sparse matrix-vector operations. Processes **9**(4) (2021)
16. Hernández-Tello, J., Martínez-Del-Amor, M.Á., Orellana-Martín, D., Cabarle, F.G.C.: Sparse spiking neural-like membrane systems on graphics processing units. International Journal of Neural Systems **34**(7) (2024) 2450038–2450038
17. Lloreda, C.L.: Wi-fi for neurons: first map of wireless nerve signals unveiled in worms. Nature **623**(7989) (2023) 894–895
18. Randi, F., Sharma, A.K., Dvali, S., Leifer, A.M.: Neural signal propagation atlas of caenorhabditis elegans. Nature (2023) 1–9
19. Ripoll-Sánchez, L., Watteyne, J., Sun, H., Fernandez, R., Taylor, S.R., Weinreb, A., Bentley, B.L., Hammarlund, M., Miller, D.M., Hobert, O., Beets, I., Vértes, P.E., Schafer, W.R.: The neuropeptidergic connectome of c. elegans. Neuron **111**(22) (2023) 3570–3589.e5
20. Wang, H., Qian, T., Zhao, Y., Zhuo, Y., Wu, C., Osakada, T., Chen, P., Chen, Z., Ren, H., Yan, Y., Geng, L., Fu, S., Mei, L., Li, G., Wu, L., Jiang, Y., Qian, W., Zhang, L., Peng, W., Xu, M., Hu, J., Jiang, M., Chen, L., Tang, C., Zhu, Y., Lin, D., Zhou, J.N., Li, Y.: A tool kit of highly selective and sensitive genetically encoded neuropeptide sensors. Science **382**(6672) (2023) eabq8173
21. Gh. Păun: Spiking neural P systems: A tutorial. Bull. Eur. Assoc. Theor. Comput. Sci. **91** (Feb 2007) 145–159
22. Gh. Păun, Rozenberg, G., Salomaa, A., eds.: The Oxford Handbook of Membrane Computing. Oxford Univeristy Press (2010)
23. Minsky, M.L.: Computation: finite and infinite machines. Prentice-Hall, Inc., USA (1967)
24. Macababayao, I.C.H., Cabarle, F.G.C., de la Cruz, R.T.A., Zeng, X.: Normal forms for spiking neural p systems and some of its variants. Information Sciences **595** (2022) 344–363

25. Cabarle, F.G.C., Buño, K.C., Adorna, H.N.: Time after time: notes on delays in spiking neural p systems. In: Theory and Practice of Computation: 2nd Workshop on Computation: Theory and Practice, Manila, The Philippines, September 2012, Proceedings, Springer (2013) 82–92
26. Freund, R., Ionescu, M., Oswald, M.: Extended spiking neural p systems with decaying spikes and/or total spiking. International Journal of Foundations of Computer Science **19**(05) (2008) 1223–1234
27. Ibarra, O.H., Woodworth, S., Yu, F., Păun, A.: On spiking neural p systems and partially blind counter machines. In: Unconventional Computation: 5th International Conference, UC 2006, York, UK, September 4-8, 2006. Proceedings 5, Springer (2006) 113–129
28. Cabarle, F.G.C., Adorna, H.N., Pérez-Jiménez, M.J.: Sequential spiking neural p systems with structural plasticity based on max/min spike number. Neural computing and applications **27** (2016) 1337–1347
29. Bibi, A., Xu, F., Adorna, H.N., Cabarle, F.G.C., et al.: Sequential spiking neural p systems with local scheduled synapses without delay. Complexity **2019** (2019)
30. Zeng, X., Adorna, H., Martínez-del Amor, M.Á., Pan, L., Pérez-Jiménez, M.J.: Matrix representation of spiking neural p systems. In: Membrane Computing: 11th International Conference, CMC 2010, Jena, Germany, August 24-27, 2010. Revised Selected Papers 11, Springer (2011) 377–391
31. Adorna, H.N.: Matrix representations of spiking neural p systems: Revisited. arXiv preprint arXiv:2211.15156 (2022)
32. Aboy, B.C.D., Bariring, E.J.A., Carandang, J.P., Cabarle, F.G.C., De La Cruz, R.T., Adorna, H.N., Martínez-del Amor, M.Á.: Optimizations in cusnp simulator for spiking neural p systems on cuda gpus. In: 2019 International Conference on High Performance Computing & Simulation (HPCS), IEEE (2019) 535–542
33. Odasco, A.N.L., Rey, M.L.M., Cabarle, F.G.C.: Improving gpu web simulations of spiking neural p systems. Journal of Membrane Computing (2023) 1–16
34. Gungon, R.V., Hernandez, K.K.M., Cabarle, F.G.C., De la Cruz, R.T.A., Adorna, H.N., Martínez-del Amor, M.Á., Orellana-Martín, D., Pérez-Hurtado, I.: Gpu implementation of evolving spiking neural p systems. Neurocomputing **503** (2022) 140–161
35. Uy, A.V.D., Wu, J.J.Q.C., Cabarle, F.G.C., de la Cruz, R.T.A., Adorna, H.N.: Evolving spiking neural p systems with rules on synapses on cuda. Philippine Computing Journal **17** (2022) 10–31
36. Jimenez, Z.B., Cabarle, F.G.C., de la Cruz, R.T.A., Buño, K.C., Adorna, H.N., Hernandez, N.H.S., Zeng, X.: Matrix representation and simulation algorithm of spiking neural p systems with structural plasticity. Journal of Membrane Computing **1** (2019) 145–160
37. Alhazov, A., Freund, R., Oswald, M., Slavkovik, M.: Extended spiking neural p systems. In Hoogeboom, H.J., Păun, G., Rozenberg, G., Salomaa, A., eds.: Membrane Computing, Berlin, Heidelberg, Springer Berlin Heidelberg (2006) 123–134
38. Verlan, S., Freund, R., Alhazov, A., Ivanov, S., Pan, L.: A formal framework for spiking neural p systems. Journal of Membrane Computing **2**(4) (2020) 355–368
39. Chen, H., Freund, R., Ionescu, M., Păun, G., Pérez-Jiménez, M.J.: On string languages generated by spiking neural p systems. Fundamenta Informaticae **75**(1-4) (2007) 141–162
40. Cabarle, F.G.C., de la Cruz, R.T.A., Zhang, X., Jiang, M., Liu, X., Zeng, X.: On string languages generated by spiking neural p systems with structural plasticity. IEEE transactions on nanobioscience **17**(4) (2018) 560–566
41. Păun, A., Păun, G.: Small universal spiking neural p systems. BioSystems **90**(1) (2007) 48–60
42. Cabarle, F.G.C., de la Cruz, R.T.A., Adorna, H.N., Dimaano, M.D., Peña, F.T., Zeng, X.: Small spiking neural p systems with structural plasticity. Enjoying Natural Computing: Essays Dedicated to Mario de Jesús Pérez-Jiménez on the Occasion of His 70th Birthday (2018) 45–56

43. de la Cruz, R.T.A., Cabarle, F.G.C., Adorna, H.N.: Steps toward a homogenization procedure for spiking neural p systems. Theoretical Computer Science **981** (2024) 114250
44. de la Cruz, R.T.A., Cabarle, F.G.C., Macababayao, I.C.H., Adorna, H.N., Zeng, X.: Homogeneous spiking neural p systems with structural plasticity. Journal of Membrane Computing **3** (2021) 10–21
45. Cabarle, F.G.C., Hernandez, N.H.S., Martínez-del Amor, M.Á.: Spiking neural p systems with structural plasticity: Attacking the subset sum problem. In: International Conference on Membrane Computing, Springer (2015) 106–116
46. Paul, P., Sosik, P.: Solving the sat problem using spiking neural p systems with coloured spikes and division rules. (2024)
47. Ishdorj, T.O., Leporati, A.: Uniform solutions to sat and 3-sat by spiking neural p systems with pre-computed resources. Natural Computing **7** (2008) 519–534
48. Idowu, R.K., Chandren, R., Othman, Z.A.: Advocating the use of fuzzy reasoning spiking neural p system in intrusion detection. In: Asian Conference on Membrane Computing ACMC 2014. (2014) 1–5
49. Song, T., Pang, S., Hao, S., Rodríguez-Patón, A., Zheng, P.: A parallel image skeletonizing method using spiking neural p systems with weights. Neural Processing Letters **50** (2019) 1485–1502