Spiking Neural P Systems with Weights and Thresholds

Jun Wang[1], Hendrik Jan Hoogeboom[2], Linqiang Pan[1,3⋆], Gheorghe Păun[3,4]

[1] Key Laboratory of Image Processing and Intelligent Control
Department of Control Science and Engineering
Huazhong University of Science and Technology
Wuhan 430074, Hubei, China
junwangjf@gmail.com, lqpan@mail.hust.edu.cn
[2] Leiden Institute of Advanced Computer Science, Universiteit Leiden
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
hoogeboom@liacs.nl
[3] Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
[4] Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucureşti, Romania
george.paun@imar.ro, gpaun@us.es

Summary. A variant of spiking neural P systems is introduced, with (positive or negative) weights on synapses and with the restriction that the rules of a neuron fires when the potential of that neuron equals a given threshold. The involved numbers – weights, thresholds, potential consumed by each rule – can be real (computable) numbers, rational, integer, natural numbers. The power of the obtained systems is investigated. For instance, it is shown that integer numbers (very restricted: $1, -1$ for weights, 1 and 2 for thresholds and for writing the rules) suffice in order to compute all Turing computable sets of numbers, both in the generative and the accepting modes. Using only natural numbers we characterize the family of semilinear sets of numbers. Some open problems and suggestions for further research are formulated.

1 Introduction

Spiking neural P systems (SN P systems, for short) were introduced in [5] in the aim of defining computing models based on ideas specific to spiking neurons, currently much investigated in neural computing (see, e.g., [4], [7], [8]). The resulting models are a variant of tissue-like and neural-like P systems from membrane computing – we refer to [10] for basic information in this research area, to [11] for a

⋆ Corresponding author

comprehensive presentation, and to the web site [13] for the up-to-date information.

In short, an SN P system consists of a set of *neurons* placed in the nodes of a directed graph and sending signals (*spikes*, denoted in what follows by the symbol a) along *synapses* (arcs of the graph). Thus, the architecture is that of a tissue-like P system, with only one kind of objects present in the cells. The objects evolve by means of *spiking rules*, which are of the form $E/a^c \to a; d$, where E is a regular expression over $\{a\}$ and c, d are natural numbers, $c \geq 1, d \geq 0$. The meaning is that a neuron containing k spikes such that $a^k \in L(E), k \geq c$, can consume c spikes and produce one spike, after a delay of d steps. This spike is sent to all neurons to which a synapse exists outgoing from the neuron where the rule was applied. There also are *forgetting rules*, of the form $a^s \to \lambda$, with the meaning that $s \geq 1$ spikes are forgotten, provided that the neuron contains exactly s spikes. The system works in a synchronized manner, i.e., in each time unit, the rule to be applied is non-deterministically chosen, each neuron which can use a rule should do it, but the work of the system is sequential in each neuron: only (at most) one rule is used in each neuron. One of the neurons is considered to be the *output neuron*, and its spikes are also sent to the environment. The moments of time when a spike is emitted by the output neuron are marked with 1, the other moments are marked with 0. This binary sequence is called the *spike train* of the system – it might be infinite if the computation does not stop. The *result of a computation* is encoded in the distance between consecutive spikes sent into the environment by the (output neuron of the) system.

In SN P systems, the applicability of each rule is determined by checking the content of the neuron against a regular set associated with the rule. There is a considerable computational power hidden into the implicit mechanism that SN P systems use to decide whether a given rule can be applied or not. For instance, in [6] it is proved that deciding whether a rule can be applied is at least **NP**-complete.

In this paper, a variant of SN P systems is presented, aiming to decide in an easy way the applicability of rules. To this aim, we do not count spikes, as in usual SN P systems, but we consider that each neuron contains a *potential*, which, in the general case, can be expressed by a real number (to avoid any complication, in what follows we always use computable real numbers). Each neuron fires when its potential is equal to a given *threshold*; at that time, part of the potential is consumed and a unit potential is produced (a spike). This unit potential passes to neighboring neurons multiplied with the *weights* of synapses. The weights can also be real numbers, hence both positive and negative. In this way, we can define computations and the result of computations as usual in SN P systems (the result is associated with the spike train of the computation – here we consider the distance between the first two spikes which leave the output neuron; SN P systems working in the accepting mode are also considered).

An important convention is assumed: when the potential of a neuron is higher than its firing threshold, then the potential remains unchanged (can be changed – increased or decreased – by adding new amounts coming from other neurons,

amounts which can be positive or negative, depending on synapses weights), but when the potential of a neuron is smaller than the firing threshold, then this potential vanishes, the potential of the neuron is set to zero. These assumptions are essentially used in the proofs below, but we do not know what happens when, for instance, potentials smaller than the firing thresholds remain unchanged instead of being removed.

As we will see, SN P systems with integer values for weights and potentials are computationally universal, and the proofs are rather simple (and they use very small numbers, only $1, -1$ as weights, and 1, 2 for writing the rules).

Besides the above computer science motivation, considering SN P systems with weights and firing thresholds has also a biological motivation. Like most other cells in the body, the plasma membrane of excitable cells exhibits a *membrane potential* (an electrical voltage difference across the membrane), called *resting membrane potential*, and its typical value is -70 mV. Moreover, each neuron has its own *threshold potential* which is the membrane potential to which a membrane must be depolarized to initiate an *action potential*. If the membrane potential of a neuron equals its threshold potential, then the neuron will fire, sending out an action potential (signal), and its membrane potential will return to the resting membrane potential. If the membrane potential is smaller than the threshold potential, then no signal is emitted and the membrane potential will also return to the resting membrane potential. For more details see [4] and [8].

Let us note that SN P systems with synapses which transmit negative amounts of spikes to the destination neurons were investigated, e.g., in [1], [2], where also further biological motivations can be found.

In what follows, the reader is assumed to have some familiarity with (basic elements of) language theory, e.g., from [12], as well as basic membrane computing [10] (for more updated information about membrane computing, please refer to [13]). We here mention that by $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}_c$ we denote the sets of natural, integer, rational, and computable real numbers, respectively, while *SLIN, NRE* denote the families of semilinear and of Turing computable sets of numbers. (Note that $SLIN$ is the family of length sets of regular languages and NRE is the family of length sets of recursively enumerable languages.)

Convention: when evaluating or comparing the power of two number generating/accepting devices, number zero is ignored.

2 Spiking Neural P Systems with Weights and Firing Thresholds

We introduce directly the type of spiking neural P systems which we investigate in this paper; the reader is assumed familiar with the basic elements of "classic" SN P systems.

An *SN P system with weights and thresholds* (from now on we will deal only with such systems, hence sometimes we say shortly SN P systems; when necessary

to stress the new type of systems, we will write WTSN P systems), of degree $m \geq 1$, is a construct of the form

$$\Pi = (\sigma_1, \ldots, \sigma_m, syn, in, out), \text{ where:}$$

1. $\sigma_1, \ldots, \sigma_m$ are *neurons*, of the form $\sigma_i = (p_i, R_i), 1 \leq i \leq m$, where:
   a) $p_i \in \mathbb{R}_c$ is the *initial potential* in neuron σ_i;
   b) R_i is a finite set of *spiking rules* of the from $T_i/d_j \to 1$, $j = 1, 2, \ldots, n_i$ for some $n_i \geq 1$, where $T_i \in \mathbb{R}_c$, $T_i \geq 1$, is the firing *threshold potential* of neuron σ_i, and $d_j \in \mathbb{R}_c$ with the restriction $0 < d_j \leq T_i$;
2. $syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\} \times \mathbb{R}_c$ are *synapses* between neurons, where $i \neq j$, $r \neq 0$ for each $(i, j, r) \in syn$;
3. $in, out \in \{1, 2, \ldots, m\}$ indicate the *input* and *output* neurons, respectively.

The spiking rules are applied as follows. Assume that at a given moment, neuron σ_i has the potential equal to p. If $p = T_i$, then any rule $T_i/d_j \to 1 \in R_i$ can be applied. The execution of this rule consumes an amount of d_j of the potential (thus leaving the potential $T_i - d_j$) and prepares one unit potential (we also say a spike) to be delivered to all the neurons σ_j such that $(i, j, r) \in syn$. Specifically, each of these neurons σ_j receives a quantity of potential equal to r, which is added to the existing potential in σ_j. Note that r can be positive or negative, hence the potential of the receiving neuron is increased or decreased. The potential emitted by a neuron σ_i passes immediately to all neurons σ_j such that $(i, j, r) \in syn$, that is, the transition of potential takes no time. If a neuron σ_i spikes and it has no outgoing synapse, then the potential emitted by neuron σ_i is lost.

We stress that (1) each neuron σ_i has only one fixed threshold potential T_i; (2) if a neuron has the potential equal to its threshold potential, then all rules associated with this neuron are enabled, and only one of them is non-deterministically chosen to be applied; (3) when a neuron spikes, there is always only one unit potential emitted.

If neuron σ_i has the potential p such that $p < T_i$, then the neuron σ_i returns to the resting potential 0. If neuron σ_i has the potential p such that $p > T_i$, then the potential p keeps unchanged.

Summing up, if neuron σ_i has potential p and receives potential k at step t, then at step $t + 1$ it has the potential p', where:

$$p' = \begin{cases} k, & \text{if } p < T_i; \\ p - d_j + k, & \text{if } p = T_i \text{ and rule } T_i/d_j \to 1 \text{ is applied;} \\ p + k, & \text{if } p > T_i. \end{cases}$$

As usual in membrane computing, a global clock is assumed, marking the time for the whole system, hence the functioning of the system is synchronized. Each neuron uses at most one rule in each step, non-deterministically chosen among its rules, provided that its potential equals the firing threshold, but all neurons which can use a rule must do it.

The *configuration* of the system is described by the distribution of potentials in neurons. The *initial configuration* of the system is the tuple $\langle p_1, \ldots, p_m \rangle$. Using the rules as suggested above, we can define transitions among configurations. Any sequence of transitions starting from the initial configuration is called a *computation*. A computation halts if it reaches a configuration where no rule can be used. With any computation, halting or not, we associate a *spike train*, the binary sequence with occurrences of 1 indicating time instances when the output neuron sends one unit potential (a spike) out of the system (we also say that the system itself spikes at that time).

The result of a computation can be defined in several ways. In this paper, with any spike train containing at least two spikes, the first two being emitted at step t_1, t_2, one associates a result, in the form of the number $t_2 - t_1$; we say that this number is computed by Π. The set of all numbers computed in this way by Π is denoted by $N_2(\Pi)$ (the subscript indicates that we only consider the distance between the first two spikes of any computation; note that 0 cannot be computed, that is why we disregard this number when investigating the computing power of any device).

SN P systems can also work in the accepting mode: we start the computation from the initial configuration, and we introduce in the input neuron two spikes, in steps t_1 and t_2 (hence we introduce in σ_{in} one unit of potential in each step t_1 and t_2); the number $t_2 - t_1$ is accepted by the system if the computation eventually halts. We denote by $N_{acc}(\Pi)$ the set of numbers accepted by Π.

In the generative case, the neuron with label *in* is ignored; in the accepting mode, the neuron with label *out* is ignored (sometimes below, we identify the neuron σ_i with its label i, so we say "neuron i" understanding that we speak about "neuron σ_i").

We denote by $N_\alpha WT_X SNP_m$ the families of all sets $N_\alpha(\Pi)$, $\alpha \in \{2, acc\}$, computed by WTSN P systems with at most $m \geq 1$ neurons, using weights, thresholds, and amounts of consumed potentials in the rules taken from the set X, for $X \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}_c\}$. When the number of neurons is not bounded, the subscript m is replaced with $*$.

Usually, in the SN P systems area one takes into account several other parameters describing the size of the used systems, such as the maximal number of rules in a neuron, the maximal number of spikes consumed by a rule, etc. Here we can also consider the maximal firing threshold, the maximal positive weight and the minimal negative weight of a synapse, etc. However, as we will see in the following sections, these parameters will have very small values in all results we obtain, so we prefer to simplify the notation and ignore these parameters.

3 One Example

In the next sections we will give several explicit constructions of SN P systems with weights and thresholds, always with integer numbers for describing the potentials

and the rules, that is why we discuss here only one example, where rational non-integer numbers are used.

As usual in this area, the systems are represented graphically, which may be easier to understand than in a symbolic way. We use an oval with the initial potential and spiking rules inside to represent a neuron, and arrows between these ovals to represent the synapses; numbers will mark these arrows, indicating the weights. The input neuron has an incoming arrow and the output neuron has an outgoing arrow, suggesting their communication with the environment. When the weight on a synapse is one, we omit writing it.

Consider the SN P system Π as shown in Figure 1, which consists of three neurons.

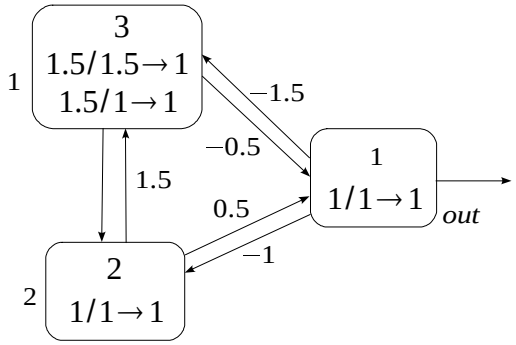

Fig. 1. Example of a WTSN P system Π

At step 1, only output neuron σ_{out} spikes, while the other two neurons σ_1, σ_2 maintain their potentials, because their potentials are greater than their corresponding firing thresholds. Neurons σ_1 and σ_2 receive potentials -1.5 and -1, respectively, from neuron σ_{out}. At step 2, neurons σ_1 and σ_2 have potentials 1.5 and 1, respectively, which equal their corresponding firing thresholds, hence both neurons σ_1 and σ_2 spike.

When neuron σ_2 spikes, it consumes one unit of potential and, at the same time, it receives one unit of potential from neuron σ_1, hence at next step it still has potential 1, and spikes again. Neuron σ_1 has two rules, $1.5/1.5 \to 1$ and $1.5/1 \to 1$, and one of them is non-deterministically chosen. If rule $1.5/1.5 \to 1$ is applied, then with consuming potential 1.5 and receiving potential 1.5 from neuron σ_2, the potential of neuron σ_1 is still 1.5, hence it will spike again. In this way, neurons σ_1 and σ_2 can spike as long as rule $1.5/1.5 \to 1$ is chosen to be applied. During this process, at each step, neuron σ_{out} receives potential -0.5 from σ_1 and potential 0.5 from σ_2, which means that neuron σ_{out} has potential 0 and does not spike.

If at step $t \geq 2$, rule $1.5/1 \to 1$ is chosen to be applied, then at step $t+1$, neuron σ_1 has the potential $1.5 - 1 + 1.5 = 2$, which is greater than its threshold

and it will not spike. At step $t+1$, neuron σ_2 has potential 1 and spikes; neuron σ_{out} receives potential 1 from neuron σ_2 at step $t+1$ and spikes at step $t+2$. At step $t+2$, neuron σ_1 receives potential -1.5 from neuron σ_{out}, its potential changes to $2-1.5=0.5$, which is less than its threshold potential 1.5 and the neuron returns to resting potential 0 at step $t+3$. At step $t+2$, neuron σ_2 receives potential -1 from neuron σ_{out}, its potential changes to $0-1=-1$, which is less than its threshold potential 1 and returns to resting potential 0 at step $t+3$. So, the system halts after step $t+3$.

The number generated is $(t+2)+1=t+3$, where $t \geq 2$, and the value of t depends on the non-deterministic choice of rules $1.5/1.5 \to 1$ or $1.5/1 \to 1$ in neuron σ_1. Thus, $N_2(\Pi) = \mathbb{N} - \{1,2\}$ (recall that the number 0 is ignored when investigating the computational power of devices).

4 Preliminary Results

Let us start by noting some immediate relations, following from the definitions:

Lemma 1. *(i)* $N_\alpha WT_{\mathbb{N}} SNP_m \subseteq N_\alpha WT_{\mathbb{Z}} SNP_m \subseteq N_\alpha WT_{\mathbb{Q}} SNP_m \subseteq N_\alpha WT_{\mathbb{R}_c} SNP_m \subseteq NRE$, *for all* $\alpha \in \{2, acc\}$ *and* $m \geq 1$ *or* $m = *$.

(ii) $N_\alpha WT_{\mathbb{X}} SNP_m \subseteq N_\alpha WT_{\mathbb{X}} SNP_{m'} \subseteq N_\alpha WT_{\mathbb{X}} SNP_*$, *for all* $\alpha \in \{2, acc\}$, $m' \geq m \geq 1$, *and* $X \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}_c\}$.

All relations are obvious, with the inclusion $N_\alpha WT_{\mathbb{R}_c} SNP_m \subseteq NRE$ being a consequence of the fact that we use only computable real numbers and of Turing-Church thesis.

For a given WTSN P system $\Pi = (\sigma_1, \ldots, \sigma_m, syn, in, out)$ and a constant $k \in \mathbb{R}_c$, let us denote by $k\Pi$ the system obtained by multiplying by k all weights and potentials from Π (if a rule of Π is of the form $T_i/d_j \to 1$, then in $k\Pi$ we use the rule $kT_i/kd_j \to 1$; a synapse (i,j,r) will become (i,j,kr) in $k\Pi$, hence transporting a potential equal to kr when neuron σ_i produces one spike).

Lemma 2. *For any WTSN P system* Π *and constant* $k \in \mathbb{R}_c$, *with* $k\Pi$ *constructed as above, we have* $N_\alpha(\Pi) = N_\alpha(k\Pi)$, *for all* $\alpha \in \{2, acc\}$.

The assertion directly follows from the way $k\Pi$ is defined, and has the next interesting consequence:

Corollary 1. $N_\alpha WT_{\mathbb{Z}} SNP_m = N_\alpha WT_{\mathbb{Q}} SNP_m$ *for all* $\alpha \in \{2, acc\}$ *and* $m \geq 1$ *or* $m = *$.

One inclusion is pointed out in Lemma 1, the opposite one follows from Lemma 2: take an arbitrary WT SN P systems with all constants in $\mathbb{Q}$, let k be the least common multiple of all denominators of rational numbers in Π (weights and potentials), and consider $k\Pi$. This has all used numbers integers and it is equivalent with Π.

5 Universality of WTSN P Systems with Integer Numbers

Expected enough, we obtain universality both in the generative and the accepting case. However, these results cannot be obtained as particular cases of universality results known for usual SN P systems: the weights and thresholds bring additional possibilities to "program" the computation of an SN P system, but instead we are very much restricted by the fact that all the rules of a neuron are enabled at the same time, when the firing threshold is reached; this corresponds to usual SN P systems with a finite number of spikes in each neuron (and only one regular expression – identifying a singleton!), and such systems are known to only compute semilinear sets of numbers.

5.1 The Generative Case

Consider first the case of sets $N_2(\Pi)$; we have the following result:

Theorem 1. $N_2WT_{\mathbb{Z}}SNP_* = NRE$.

Proof. We only have to prove the inclusion $NRE \subseteq N_2WT_{\mathbb{Z}}SNP_*$, and to this aim we use the characterization of NRE by means of register machines used in the generating mode.

Let us consider a register machine $M = (m, H, l_0, l_h, I)$ (number of registers, set of labels, initial label, halt label, set of instructions) which is assumed that in the halting configuration has all registers different from the first one empty, and that output register is never decremented during the computation. We construct a WTSN P system Π to simulate M as follows. We construct modules ADD and SUB to simulate the instructions of M, as well as an output module FIN which provides the result (in the form of a suitable spike train). Each register r of M will have a neuron σ_r in Π, and if the register contains the number n, then the associated neuron will have the potential $2n + 2$. A neuron σ_{l_i} is associated with each label $l_i \in H$, and some auxiliary neurons $\sigma_{l_i^{(j)}}$, $j = 1, 2, 3, \ldots$, will also be considered, thus precisely identified by label l_i (remember that each $l_i \in H$ is associated with a unique instruction of M).

The modules will be given in a graphical form. In the initial configuration, all neurons have the potential 0, except that the neuron associated with label l_0 of M has potential 1 and the neurons associated with the registers have potential 2. In general, when a neuron σ_{l_i}, where $l_i \in H$, has potential 1, then that neuron becomes active and the module associated with the respective instruction of M starts to work, simulating the instruction.

Module ADD – simulating an ADD instruction $l_i : (\texttt{ADD}(r), l_j, l_k)$.

Module ADD, shown in Figure 2, is composed of eight neurons: neuron σ_r for register r, neurons $\sigma_{l_i}, \sigma_{l_j}, \sigma_{l_k}$ for instructions with labels l_i, l_j, l_k, and four auxiliary neurons.

The initial instruction of M, the one with label l_0, is an ADD instruction. Let us assume that at step t we have to simulate an instruction $l_i : (\texttt{ADD}(r), l_j, l_k)$,

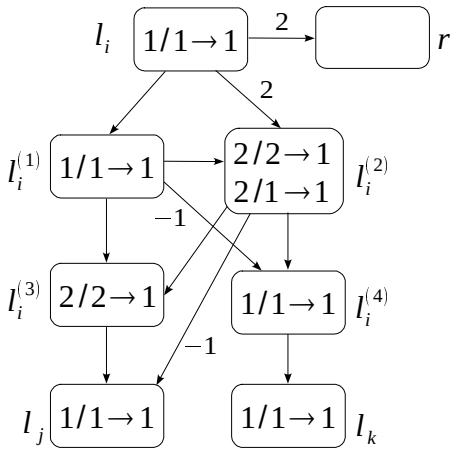

Fig. 2. Module ADD (simulating $l_i : (\texttt{ADD}(r), l_j, l_k)$)

with neuron σ_{l_i} having potential 1 and other neurons having resting potential 0, except those neurons associated with registers. Having potential 1, neuron σ_{l_i} fires by rule $1/1 \to 1$. Simultaneously, neurons $\sigma_{l_i^{(1)}}$, $\sigma_{l_i^{(2)}}$, and σ_r receive potentials 1, 2, 2, respectively. In this way, the potential of neuron σ_r increased by two, thus simulating the increase of the number stored in register r by one.

At the next step, the computation of M passes non-deterministically to one of the instructions with labels l_j and l_k; that is, we have to ensure the firing of neurons σ_{l_j} or σ_{l_k} in system Π, non-deterministically choosing one of them. To this aim, we use the non-deterministic choice of rules $2/2 \to 1$ and $2/1 \to 1$ in $\sigma_{l_i^{(2)}}$. Because neuron $\sigma_{l_i^{(2)}}$ has potential 2 (received from neuron σ_{l_i} at the last step), it has to choose non-deterministically one of these rules. We have two cases.

(1) If rule $2/2 \to 1$ is applied at step $t+1$, then neuron $\sigma_{l_i^{(2)}}$ consumes its potential for spiking. With receiving potential 1 from neuron $\sigma_{l_i^{(1)}}$ at step $t+1$, neuron $\sigma_{l_i^{(2)}}$ has potential 1 at step $t+2$, which is less than its threshold potential, hence the neuron returns to the resting potential 0. At step $t+1$, neuron σ_{l_j} receives potential -1 from neuron $\sigma_{l_i^{(2)}}$, which is less than its firing threshold and it returns to the resting potential 0 at step $t+2$. At step $t+1$, neuron $\sigma_{l_i^{(4)}}$ receives potential -1 from neuron $\sigma_{l_i^{(1)}}$ and potential 1 from neuron $\sigma_{l_i^{(2)}}$, hence its potential is still 0. At step $t+1$, neuron $\sigma_{l_i^{(3)}}$ receives potential 2 from neurons $\sigma_{l_i^{(1)}}$ and $\sigma_{l_i^{(2)}}$ and at step $t+2$ it spikes by rule $2/2 \to 1$. Receiving potential 1 from neuron $\sigma_{l_i^{(3)}}$, neuron σ_{l_j} becomes active, starting to simulate the instruction l_j of M.

(2) If rule $2/1 \to 1$ is applied at step $t+1$, then neuron $\sigma_{l_i^{(2)}}$ consumes one unit of its potential for spiking. With receiving potential 1 from neuron $\sigma_{l_i^{(1)}}$ at step

$t+1$, neuron $\sigma_{l_i^{(2)}}$ still has potential 2 at step $t+2$, and spikes again. At step $t+1$, neuron σ_{l_j} receives potential -1 from neuron $\sigma_{l_i^{(2)}}$, which is less than its threshold potential and returns to the resting potential 0 at step $t+2$. At step $t+2$, neuron σ_{l_j} receives potential -1 from neuron $\sigma_{l_i^{(2)}}$ and potential 1 from neuron $\sigma_{l_i^{(3)}}$, so its potential is zero. At step $t+1$, neuron $\sigma_{l_i^{(4)}}$ receives potential -1 from neuron $\sigma_{l_i^{(1)}}$ and potential 1 from neuron $\sigma_{l_i^{(2)}}$, so its potential remains 0. At step $t+2$, neuron $\sigma_{l_i^{(4)}}$ receives one unit of potential from neuron $\sigma_{l_i^{(2)}}$ and it spikes at step $t+3$. Receiving potential 1 from neuron $\sigma_{l_i^{(4)}}$ at step $t+3$, neuron σ_{l_k} becomes active, starting to simulate the instruction l_k of M.

Therefore, from firing neuron σ_{l_i}, we pass to firing non-deterministically one of neurons σ_{l_j}, σ_{l_k}, which correctly simulates the ADD instruction $l_i : (\texttt{ADD}(r), l_j, l_k)$.

Module SUB – simulating a SUB instruction $l_i : (\texttt{SUB}(r), l_j, l_k)$

Module SUB, shown in Figure 3, is composed of seven neurons: neuron σ_r for register r, neurons $\sigma_{l_i}, \sigma_{l_j}, \sigma_{l_k}$ for instructions with labels l_i, l_j, l_k, and three auxiliary neurons $\sigma_{l_i^{(1)}}, \sigma_{l_i^{(2)}}, \sigma_{l_i^{(3)}}$.

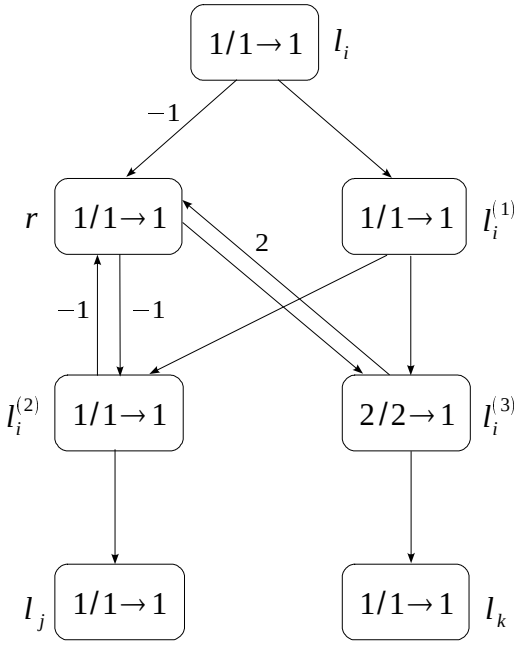

Fig. 3. Module SUB (simulating instruction $l_i : (\texttt{SUB}(r), l_j, l_k)$

Instruction l_i is simulated in Π in the following way. Initially, neuron σ_{l_i} has potential 1, and other neurons have potential 0, except neurons associated with

registers. Let t be the moment when neuron σ_{l_i} fires. At step t, neurons $\sigma_{l_i^{(1)}}$ and σ_r receive potentials 1 and -1, respectively. At step $t+1$, neurons $\sigma_{l_i^{(1)}}$ fires, neurons $\sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$ receive potential 1 from neuron $\sigma_{l_i^{(1)}}$. For neuron σ_r, there are the following two cases.

(1) The potential of neuron σ_r is 2 at step t (that is, the number stored in register r is 0). Then, at step $t+1$, neuron σ_r has potential 1 (it has received potential -1 from neuron σ_{l_i} at the previous step), and it spikes by rule $1/1 \to 1$. At step $t+1$, neuron $\sigma_{l_i^{(2)}}$ receives potential -1 from neuron σ_r and potential 1 from neuron $\sigma_{l_i^{(1)}}$, so it has potential 0. At step $t+1$, neuron $\sigma_{l_i^{(3)}}$ receives potential 2 (one unit of potential from neuron σ_r, another one from neuron $\sigma_{l_i^{(1)}}$), and it spikes at step $t+2$. Receiving potential 1 from neuron $\sigma_{l_i^{(3)}}$, neuron σ_{l_k} becomes active, and start to simulate the instruction l_k of M. Note that at step $t+2$, neuron σ_r receives potential 2 from neuron $\sigma_{l_i^{(3)}}$, and in this way, it correctly ends with potential 2, which corresponds to the fact that the number stored in register r is 0.

(2) The potential of neuron σ_r is $2n+2$ ($n > 0$) at step t. Then, at step $t+1$, neuron σ_r has potential $2n+1$, which is greater than its threshold, and will keep unchanged. At step $t+1$, neuron $\sigma_{l_i^{(3)}}$ receives potential 1 from neuron $\sigma_{l_i^{(1)}}$, which is less than its threshold potential, hence it will not spike and have potential 0 at step $t+2$. At step $t+1$, neuron $\sigma_{l_i^{(2)}}$ receives potential 1 from neuron $\sigma_{l_i^{(1)}}$, and it spikes at step $t+2$. Receiving potential -1 from neuron $\sigma_{l_i^{(2)}}$ at step $t+2$, the potential of neuron σ_r changes to $2n$, and in this way, it simulates that the number stored in register r is decreased by one. Receiving potential 1 from neuron $\sigma_{l_i^{(2)}}$ at step $t+2$, neuron σ_{l_j} becomes active, and starts to simulate the instruction l_j of M.

The simulation of SUB instruction is correct, we started from σ_{l_i} and ended in σ_{l_j} (if the register r is not empty and decreased by one), or in σ_{l_k} (if the register is empty).

Note that there is no interference between neurons used in the ADD 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 neurons in two SUB modules. Specifically, if there are several SUB instructions l_t which act on register r, then neurons $\sigma_{l_t^{(2)}}$ and $\sigma_{l_t^{(3)}}$ receive potentials -1 and 1 from neuron σ_r, respectively, while simulating the instruction $l_i : (\texttt{SUB}(r), l_j, l_k)$. After receiving these potentials, neurons $\sigma_{l_t^{(2)}}$ and $\sigma_{l_t^{(3)}}$ have potentials that are less than their corresponding firing thresholds, so both of them return to resting potential 0 at next step. Consequently, the interference among SUB modules will not cause undesired steps in Π.

Module FIN – outputting the result of the computation.

Module FIN is shown in Figure 4. Assume that the computation in M halts, which means that the halting instruction is reached. This means that neuron σ_{l_h} in Π has potential 1 and fires by rule $1/1 \to 1$. At that moment, neuron σ_1 has

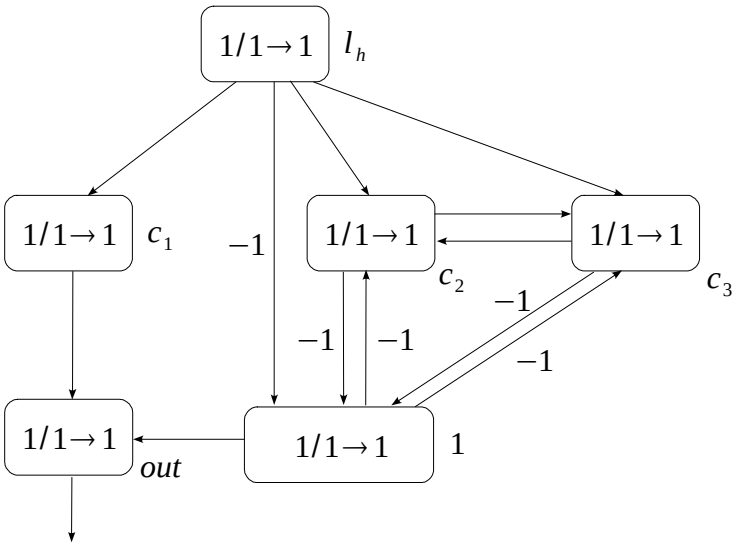

Fig. 4. Module FIN (outputting the result of computation)

potential $2n+2$, for the number $n \geq 1$ stored in register 1 of M. When σ_{l_h} fires, each neuron σ_{c_1}, σ_{c_2}, σ_{c_3} receives potential 1; neuron σ_1 receives potential -1, changing its potential to $2n+1$; suppose that this is step t. At step $t+1$, neuron σ_{c_1} spikes; neuron σ_{out} receives potential 1 from neuron neuron σ_{c_1}, and spikes at step $t+2$ (this is the first spike sent out by system Π).

From step $t+1$ on, consuming one unit potential, neurons σ_{c_2} and σ_{c_3} send potential 1 to each other, and this process continues until they receive potential -1 from neuron σ_1. During this process, at each step, neuron σ_1 receives potential -1 from neuron σ_{c_2} and -1 from σ_{c_3}, which corresponds to decreasing by one the value of the register 1. At step $t+n+1$, neuron σ_1 has potential 1 and spikes; neurons σ_{c_2} and σ_{c_3} has potential 0 after receiving potential -1 from neuron σ_1. Receiving potential 1 from σ_1 at step $t+n+1$, neuron σ_{out} spikes again at step $t+n+2$, the system sends the second spike to environment. The interval between these two spikes sent out by the system is $(t+n+2)-(t+2)=n$, which is exactly the number stored in register 1 of M at the moment when the computation of M halts.

Note that after system Π sends out the second spike, all neurons in Π have potential 0 except that neurons σ_i $(i=2,3,\ldots,m)$ have potential 2. For mathematical elegance, we can return the potentials of neurons σ_i $(i=2,3,\ldots,m)$ to 0 when the computation of Π halts. To this aim, we just need to add synapses $(out,i,-2)$ $(i=2,3,\ldots,m)$ in system Π.

If the number stored in register 1 is 0 when register machine M halts, then at step $t+1$, neuron σ_{out} has potential 2, which is greater than its threshold potential 1. In this case, neuron σ_{out} is blocked, and system Π sends no spike

to the environment. Furthermore, 0 is ignored when we investigate the power of devices.

From the above description of the modules and their work, it is clear that the register machine M is correctly simulated by the system Π. Therefore, $N_2(\Pi) = N(M)$ and this completes the proof. □

Let us now examine the weights used in the previous proof. In the ADD module we have two synapses with weight 2. This value can be avoided, so that the module only uses weights 1 and -1 in the following way: consider two new neurons, say σ_a and σ_b, intermediate between σ_{l_i} and its neighboring neurons (specifically, with synapses $(l_i, a, 1), (l_i, b, 1), (a, r, 1), (b, r, 1), (a, l_i^{(1)}, 1), (a, l_i^{(2)}, 1), (b, l_i^{(2)}, 1)$. Each of the new neurons holds the rule $1/1 \to 1$. In this way, both σ_r and $\sigma_{l_i^{(2)}}$ get two potential units, two steps after activating σ_{l_i}, simultaneously with one potential unit coming to $\sigma_{l_i^{(1)}}$. From now on the work of the module continues as described above. The same trick can be used in the SUB module in order to remove the single synapse with weight 2, with the mentioning that now no synchronization problem appears, hence the synapse is removed and two intermediate neurons are introduced, similar to σ_a, σ_b above, between $\sigma_{l_i^{(3)}}$ and σ_r. Module FIN contains only synapses with weights 1 and -1.

Consequently, the universality is obtained with WTSN P systems of a rather restricted form. This observation deserves to be formulated as a *normal form* result:

Corollary 2. *The universality of WTSN P systems is preserved if we use only (i) weights 1 and* -1 *for synapses, (ii) at most two rules per neuron, and (iii) all rules are of one of the following three forms:* $1/1 \to 1$, $2/2 \to 1$, *and* $2/1 \to 1$.

The use of two rules in at least one neuron cannot be avoided, because in the generative case the system should be non-deterministic, otherwise we generate nothing or a singleton; non-determinism means choosing between rules, hence we need at least two in the same neuron.

5.2 The Accepting Case

The number of rules per neuron can be decreased to one in this case, due to the fact that the ADD instructions of a register machine used in the accepting mode can be taken deterministic.

The number to be computed is introduced in the system as the distance between the first two unit potentials which enter the input neuron. This is done by means of a module INPUT as indicated in Figure 5. After receiving two unit potentials, if the system halts, then the number is accepted. So, we do not need a module for outputting the result of computation. For a SUB instruction $l_i : (\texttt{SUB}(r), l_j, l_k)$, we use the same module as in Figure 3. For a deterministic ADD instruction, of the form $l_i : (\texttt{ADD}(r), l_j)$, we consider the simple module given in Figure 6. Initially, all neurons in the system have the potential 0. The way the modules work can be checked in a similar way as in the proof of Theorem 1.

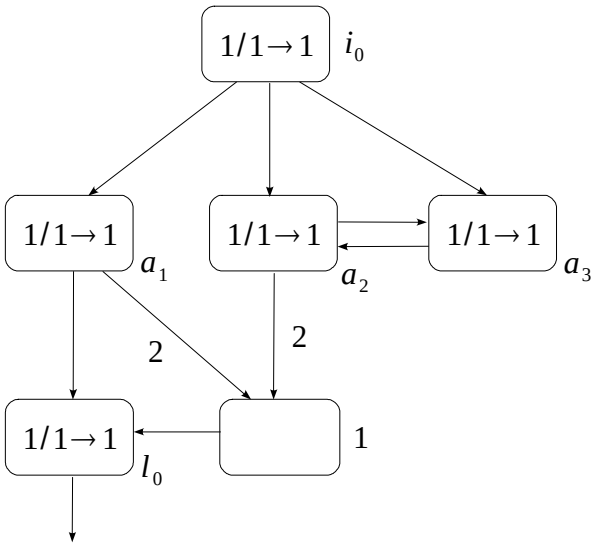

Fig. 5. Module INPUT (initializing the computation)

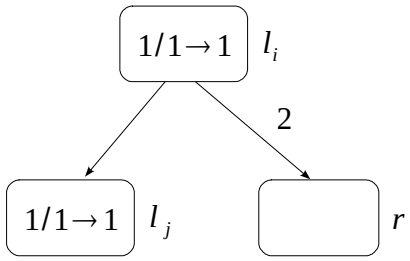

Fig. 6. Module ADD in the deterministic case

We conclude with the following counterpart of Theorem 1 (with the mentioning that also the assertions in Corollary 2 hold true, with point (ii) stating that exactly one rule per neuron is used; the removing of synapses with weight 2 is done in the same way as described above):

Theorem 2. $N_{acc}WT_{\mathbb{Z}}SNP_* = NRE$.

The previous results can be summarized as follows:

Corollary 3. $N_\alpha WT_{\mathbb{N}}SNP_m \subseteq N_\alpha WT_{\mathbb{Z}}SNP_m = N_\alpha WT_{\mathbb{Q}}SNP_m = N_\alpha WT_{\mathbb{R}_c}SNP_m = NRE$, *for all* $\alpha \in \{2, acc\}$ *and* $m \geq 1$ *or* $m = *$.

6 Systems with Natural Numbers as Weights and Thresholds

The only case which has remained unsettled is that when our systems use natural numbers as weights, thresholds, and potentials. Somewhat surprising at the first sight, such systems characterize the family of semilinear sets. For the proof of this assertion we use a series of lemmas.

Lemma 3. *Every finite set of natural numbers is in the family* $N_2WT_{\mathbb{N}}SNP_*$.

Proof. Let us take a finite set of natural numbers, $U = \{n_1, n_2, \ldots, n_k\}$, all of them different from zero. We construct a WTSN P system as suggested in Figure 7. Specifically, for each number n_i we have a "subsystem" composed of neurons $\sigma_{(i,a)}, \sigma_{(i,b)}, \sigma_{(i,c)}, \sigma_{(i,0)}, \sigma_{(i,1)}, \ldots, \sigma_{(i,n_i+1)}$, with synapses, rules, and initial potentials as indicated in figure. A synapse exists from neuron $\sigma_{(i,n_i+1)}$ to the output neuron, σ_{out}. There also exists one further neuron, σ_0, for which only two synapses exist, $((1,0),0,1)$ and $(0,out,1)$. Figure 7 only shows two generic subsystems, and the subsystem which helps in generating number $n_k = 1$.

This system works as follows. All neurons behave deterministically, except $\sigma_{(i,c)}$, for each $1 \leq i \leq k$. As long as such a neuron uses its second rule, $2/1 \to 1$, it can spike again in the next step: one potential unit remains inside and a further one is received from $\sigma_{(i,b)}$, hence the initial amount, equal to the firing threshold, is restored. In turn, as long as $\sigma_{(i,c)}$ spikes, the sequence of neurons $\sigma_{(i,1)}, \ldots, \sigma_{(i,n_i+1)}$ continues to work, moving to the right towards σ_{out}, the neuron which can fire: a neuron in this sequence must receive two spikes in order to fire, one from the preceding neuron (initially, $\sigma_{(i,0)}$ fires, because it has inside one potential unit), and one from $\sigma_{(i,c)}$. Note that each time unit, $\sigma_{(i,c)}$ is fed by one potential unit by $\sigma_{(i,b)}$, which works forever in cooperation with $\sigma_{(i,a)}$ (they can be stopped, e.g., when σ_{out} spikes, by sending to them a further spike, but this detail is not important for our result – it can be so for other ways of defining the output, if halting is relevant). If all neurons in the sequence $\sigma_{(i,1)}, \ldots, \sigma_{(i,n_i+1)}$ work, then in step $n_i + 2$ a spike is set to σ_{out} and in step $n_i + 3$ a spike is sent to the environment. Note however that the first spike is sent out of the system in step 3, on the path $\sigma_{(1,0)}, \sigma_0, \sigma_{out}$. Consequently, the distance between these spikes is $(n_i+3)-3 = n_i$.

However, any of these processes of sending a spike towards σ_{out} along the path $\sigma_{(i,1)}, \ldots, \sigma_{(i,n_i+1)}$ can be stopped at any step after the first one, by using the rule $2/2 \to 1$ in neuron $\sigma_{(i,c)}$: using this rule consumes both potential units of $\sigma_{(i,c)}$, only one unit is received from $\sigma_{(i,b)}$, which is removed, and $\sigma_{(i,c)}$ remains idle. Therefore, non-deterministically, we can stop all but one sequence $\sigma_{(i,1)}, \ldots, \sigma_{(i,n_i+1)}$ of neurons, $1 \leq i \leq k$, so that the output neuron receives only two spikes, the one in step 3, along the path $\sigma_{(1,0)}, \sigma_0, \sigma_{out}$, and the one along the path $\sigma_{(i,1)}, \ldots, \sigma_{(i,n_i+1)}$ which remained unblocked. In conclusion, each number in the set $\{n_1, n_2, \ldots, n_k\}$ can be generated, that is, $N_2(\Pi) = U$, hence $FIN \subseteq N_2AT_{\mathbb{N}}SNP_*$.

Of course, the number of neurons depends on the sum of numbers in the set U, but some neurons in the previous construction can be saved (a unique pair $\sigma_{(i,a)}, \sigma_{(i,b)}$ can fed up all neurons $\sigma_{(i,c)}$), but this aspect is not relevant for us. □

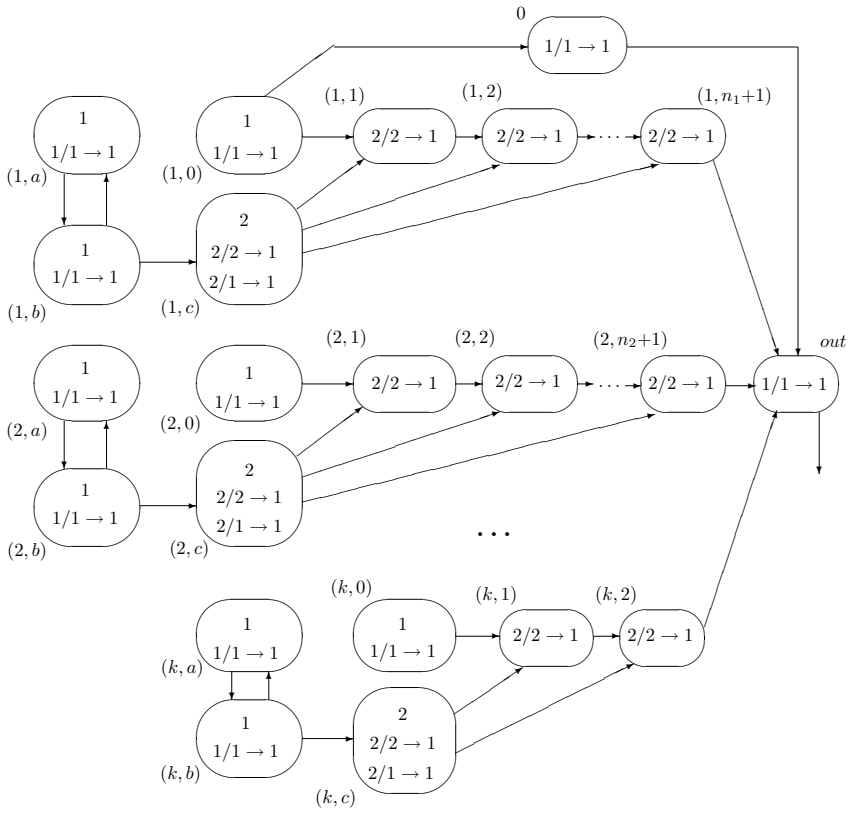

Fig. 7. A WTSN P system generating a finite set of numbers

Lemma 4. *Any arithmetical progression* $P_{k,l} = \{kn + l \mid n \geq 1\}$ *with* $k \geq 2, l \geq 2$ *is in the family* $N_2WT_{\mathbb{N}}SNP_*$.

Proof. Let us consider the system Π in Figure 8. It generates the set $N_2(\Pi) = \{2n + 2 \mid n \geq 1\}$.

The output neuron spikes in the first step and then only after receiving a spike from neuron σ_4. In turn, this neuron spikes only after receiving a spike from each neuron σ_2 and σ_3 (if we have only one spike in σ_4, then it is removed). Then, neuron σ_2 can sent a spike to σ_4 simultaneously with σ_3 only if it, after receiving two spikes from σ_2 (note that the synapse $(0, 2, 2)$ is the only one with weight 2), uses first the rule $2/1 \to 1$, so that one spike remains inside, making possible the firing in the next step. If neuron σ_2 uses the rule $2/2 \to 1$, then its spike will re-initiate the work of neuron σ_0, and the spikes from σ_2 (received from σ_1) and, after one step from σ_4, are removed. Thus, the output neuron fires for the second time after a number of passings through the cycle $\sigma_0, \sigma_1, \sigma_0$ (which means two

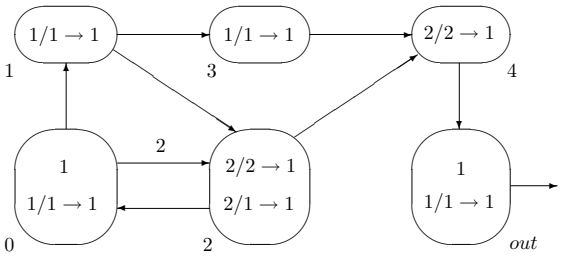

Fig. 8. A WTSN P system generating an infinite set of numbers

steps), then ending the computation, which needs two further steps. The precise checking of the functioning of the system in Figure 8 is left to the reader.

Thus, we can generate the arithmetical progression $P_{2,2}$. If we want to generate a progression $P_{k,l}$ with $k = 2 + i$ and $l = 2 + j$, then we add i neurons between σ_2 and σ_0, (instead of the synapse $(2, 0, 1)$) and j neurons between σ_4 and σ_{out}, arranged in a sequence, with the rule $1/1 \to 1$ in each of them. These neurons will lengthen the cycle $\sigma_0, \sigma_2, \ldots, \sigma_0$ with further i steps, and the path from σ_4 to σ_{out} with j steps. We denote by $\Pi_{i,j}$ the obtained system. We have $N_2(\Pi_{i,j}) = \{(2 + i)n + (2 + j) \mid n \geq 1\} = P_{k,l}$. □

Lemma 5. *If $\Pi_1, \ldots, \Pi_n$ are WTSN P systems with natural numbers as weights and potentials, and for each $1 \leq i \leq n$ there is $T_j \geq 1$ such that all computations in Π_i spike for the first time at the same step T_i, then $\bigcup_{i=1}^{n} N_2(\Pi_i) \in N_2WT_{\mathbb{N}}SNP_*$.*

Proof. Let us take separately neurons $\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_{out}$ from Figure 8, with two spikes present in σ_2 and one in σ_1 from the beginning; change also the label of σ_{out}, for instance, to σ_5, without having here any spike in the initial configuration. This system behaves like a "trigger": it sends or not a spike out of σ_5, depending on the non-deterministic behavior of σ_2.

Consider now a finite set of WTSN P systems $\Pi_1, \ldots, \Pi_n$ as in the statement of the lemma. Let T be the maximum of all $T_i, 1 \leq i \leq n$. From the output neuron of each Π_i we consider a chain of additional $T - I_i$ neurons with the unique rule $1/1 \to 1$, ending with a new output neuron. Irrespective of the length of this chain, the set of numbers generated by Π_i remains the same, as the first two spikes leaving the system remain at the same distance in time, they only leave later the system. Moreover, in this way all modified systems spikes for the first time at step T.

Take a "trigger" as above for each of the modified systems Π_i (we continue to denote by Π_i thm). Assume that a neuron σ_s from some Π_i contains $r_s \geq 1$ spikes in the initial configuration of Π_i. We remove these spikes from σ_s and establish a synapse $(5, s, r_s)$. In this way, when the neuron σ_5 of the trigger spikes, the system Π is "loaded" with exactly as many spikes as it contained initially.

In this way, non-deterministically, the "triggers" will load one or more of the systems $\Pi_i, 1 \leq i \leq n$. Take now an additional neuron which will be considered the output neuron of the whole system, let us call it σ_f, and connect all output neurons

of systems Π_i to it. With a delay of one step, the spike train of each $\Pi_i, 1 \leq i \leq n$, is produced by the new system. It is important that all systems Π_i spike for the first time at the same moment: only one of the "triggers" has to activate a system Π, all other systems $\Pi_j, j \neq i$, should remain idle, without any spike inside, otherwise the system produces no output. Indeed, if two spikes arrive at the same time in neuron σ_f (this is the case if two systems Π_i, Π_j were activated), then σ_f is blocked forever, its potential is higher than its firing threshold.

Consequently, the system whose construction was suggested above generates the union of sets $N_2(\Pi_i), 1 \leq i \leq n$. □

Theorem 3. $N_2WT_{\mathbb{N}}SNP_* = SLIN$.

Proof. (i) In order to obtain the inclusion $SLIN \subseteq N_2WT_{\mathbb{N}}SNP_*$ we use the known fact that any semilinear set is a finite union of a finite set with a finite number of arithmetical progressions. From the previous lemmas we know that finite sets and arithmetical progressions of the form $P_{k,l}$ with $k \geq 2$ and $l \geq 2$ are in $N_2WT_{\mathbb{N}}SNP_*$. Let us note that the systems constructed in the proofs of Lemmas 3 and 4 have the property in the statement of Lemma 5, to have all computations spiking for the first time at the same step. What remains to show is that also arithmetical progressions which are not of the form $P_{k,l}$ with $k \geq 2$ and $l \geq 2$ are also in $N_2WT_{\mathbb{N}}SNP_*$.

Such progressions are $P_{2,1}, P_{2,0}$, and $P_{1,l}$ for all $l \geq 0$. However, we have

$$P_{2,1} = \{3\} \cup P_{2,3}, \quad P_{2,0} = \{2\} \cup P_{2,2},$$

consequently, with Lemmas 4 and 5, they belong to $N_2WT_{\mathbb{N}}SNP_*$. Moreover,

$$P_{1,l} = (P_{1,l} \cap \{1,2,3\}) \cup (P_{1,l} \cap P_{2,2}) \cup (P_{l,l} \cap P_{3,2}).$$

Let $l_1 = \min(P_{1,l} \cap P_{2,2})$ and $l_2 = \min(P_{1,l} \cap P_{3,2})$. (Note that $l_1 \geq 4$ and $l_2 \geq 5$.) Then, we have

$$(P_{1,l} \cap P_{2,2}) = \{l_1\} \cup P_{2,l_1}, \quad (P_{1,l} \cap P_{3,2}) = \{l_2\} \cup P_{3,l_2}.$$

Using again Lemmas 4 and 5, we get $L_{1,l} \in N_2WT_{\mathbb{N}}SNP_*$, and this completes the proof of the inclusion $SLIN \subseteq N_2WT_{\mathbb{N}}SNP_*$.

(ii) The inclusion $N_2WT_{\mathbb{N}}SNP_* \subseteq SLIN$ is somewhat straightforward, after making the observation that, because all weights are positive, the potential accumulated in a neuron can be decreased only if it is smaller than or equal to the firing threshold of that neuron. Otherwise stated, if a neuron σ_i accumulates a potential strictly larger than T_i, then the potential remains larger than T_i forever (and no rule can be applied in σ_i). Therefore, the configurations of a system $\Pi = (\sigma_1, \ldots, \sigma_m, syn, out)$ can be described by a vector $\langle \alpha_1, \ldots, \alpha_m \rangle$ where $\alpha_i \in \{0, 1, \ldots, T_i\} \cup \{\bar{T}\}$, where $\bar{T}$ is just a symbol indicating that the potential of σ_i is strictly greater than T_i. If new amounts of potential are brought to a neuron whose content is already described by $\bar{T}$, then the same symbol will describe

that neuron at the next step. In this way, the functioning of the system can be described by a finite state device – e.g., by a regular (actually, right-linear, because we also need rules producing no terminal symbol) grammar: we start from the initial configuration (its description is the axiom of the grammar) and to each transition we associates a rule; because we have finitely many configurations, we have finitely many rules. As long as no spike is sent to the environment, no terminal is produced. When the first spike exits the system, we mark somehow the reached nonterminal, and from now on we produce a terminal symbol in each step (and we carry on the marking of nonterminals); when a second spike is produced by the output neuron, the derivation stops, we no longer introduce a nonterminal. The number of terminals produced is exactly the number generated by the system. The formal details are left to the reader. □

A similar result is expected for the case when WTSN P systems with natural numbers are used in the accepting mode.

7 Final Remarks

In this paper, a variant of SN P systems is introduced, using weighted synapses, potentials in neurons, and rules which handle these potentials under the control of given firing thresholds. The universality is obtained for integers used for representing all of these parameters, with the case of natural numbers as weights, potentials, and thresholds remaining to be further investigated.

Several other issues remain to be clarified about these devices.

First, we just ignored non-computable real numbers; which is their effect on the functioning and the computing power of WTSN P systems? What about considering as the result of a computation not a number related to the spike train produced by the system, but the potential of the output neuron in the halting configuration? In this case we compute real numbers, which is a rather new aspect in membrane computing. Is this feature useful for applications of SN P systems in learning and pattern recognition?

Returning to the definition: in the proofs above we have essentially used the fact that a neuron whose potential is strictly smaller than its firing threshold vanishes, it is reset to zero. What happens if this resetting does not hold, but the potential remains as it is – in the same way as a potential greater than the threshold remains unmodified. What about using the idea of decaying (e.g., as in [3]): the unused potential, irrespective of its size, decreases in each step with a specified amount (one unit, for instance)?

We stop concluding with the belief that SN P systems with weights and potentials deserve further research efforts.

Acknowledgements

The work of J. Wang and L. Pan was supported by National Natural Science Foundation of China (Grant Nos. 60674106, 30870826, 60703047, and 60803113),

Program for New Century Excellent Talents in University (NCET-05-0612), Ph.D. Programs Foundation of Ministry of Education of China (20060487014), Chenguang Program of Wuhan (200750731262), HUST-SRF (2007Z015A), and Natural Science Foundation of Hubei Province (2008CDB113 and 2008CDB180). The work of Gh. Păun was supported by Proyecto de Excelencia con Investigador de Reconocida Valía, de la Junta de Andalucía, grant P08 – TIC 04200.

References

1. A. Alhazov, R. Freund, M. Oswald, M. Slavkovik: Extended spiking neural P systems generating strings and vectors of non-negative integers. *Pre-proceedings of the 7th Workshop on Membrane Computing* (H.J. Hoogeboom, Gh. Păun, G. Rozenberg, eds.), WMC7, Leiden, 2006, 88–101.
2. A. Binder, R. Freund, M. Oswald, L. Vock: Extended spiking neural P systems with excitatory and inhibitory astrocytes. *Proceedings of Fifth Brainstorming Week on Membrane Computing* (M.A. Gutiérrez–Naranjo et al., eds.), Fenix Editora, Sevilla, 2007, 63–72.
3. R. Freund, M. Ionescu, M. Oswald: Extended spiking neural P systems with decaying spikes and/or total spiking. *Intern. J. Found. Computer Sci.*, 19 (2008), 1223–1234.
4. W. Gerstner, W. Kistler: *Spiking Neuron Models. Single Neurons, Populations, Plasticity.* Cambridge University Press, Cambridge, 2002.
5. M. Ionescu, Gh. Păun, T. Yokomori: Spiking neural P systems. *Fundamenta Informaticae*, 71, 2-3 (2006), 279–308.
6. A. Leporati, C. Zandron, C. Ferretti, G. Mauri: On the computational power of spiking neural P systems. *Proceedings of Fifth Brainstorming Week on Membrane Computing* (M.A. Gutiérrez–Naranjo et al., eds.), Fenix Editora, Sevilla, 2007, 227–246.
7. W. Maass: Computing with spikes. *Special Issue on Foundations of Information Processing of TELEMATIK*, 8, 1 (2002), 32–36.
8. W. Maass, C. Bishop, eds.: *Pulsed Neural Networks.* MIT Press, Cambridge, 1999.
9. M. Minsky: *Computation – Finite and Infinite Machines.* Prentice Hall, Englewood Cliffs, NJ, 1967.
10. Gh. Păun: *Membrane Computing – An Introduction.* Springer-Verlag, Berlin, 2002.
11. Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *Handbook of Membrane Computing.* Oxford University Press, Cambridge, 2010 (in press).
12. G. Rozenberg, A. Salomaa, eds.: *Handbook of Formal Languages.* 3 volumes, Springer-Verlag, Berlin, 1997.
13. The P System Web Page: `http://ppage.psystems.eu`