Proceedings of the International Workshop, Automata for Cellular and Molecular Computing, MTA SZTAKI, Budapest, pages 64 - 75, 2007.

Extended Spiking Neural P Systems with Decaying Spikes and/or Total Spiking

Rudolf Freund[1], Mihai Ionescu[2], and Marion Oswald[1]

[1]Faculty of Informatics, Vienna University of Technology
Favoritenstr. 9–11, A–1040 Vienna, Austria
{rudi,marion}@emcc.at

[2] Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
armandmihai.ionescu@urv.cat

Abstract

We consider extended variants of spiking neural P systems with decaying spikes (i.e., the spikes have a limited lifetime) and/or total spiking (i.e., the whole contents of a neuron is erased when it spikes). Although we use the extended model of spiking neural P systems, these restrictions of decaying spikes and/or total spiking do not allow for the generation or the acceptance of more than regular sets of natural numbers.

1 Introduction

Spiking neural P systems (in short SNP systems) are a growing research direction in the membrane computing community and have as core concept the idea of neural communication through electrical pulses called *spikes*, or *action potential*. It is known that the spikes of a given neuron all look alike, so the form of the action potential does not carry any information. But what matters is the timing and the number of spikes entering a (postsynaptic) neuron (for details on spiking neurons we refer to [5], [8] or [9]).

Initially defined in [7], SNP systems are represented as a graph with the neurons placed in the nodes of the graph. They are sending signals (spikes) along the *synapses* (the edges of the graph) if the *firing rules* inside each neuron can be activated. Hence, the structure is that of a tissue-like P system (e.g., see [4]) where the objects are all of the same form (an introduction to membrane computing can be found at [10] and an up-to-date information on this area is available online at [12]).

The functioning of an SNP system is rather simple. A global clock is assumed, and in each time unit each neuron which can use a rule should use it. The system is synchronized but it works sequentially at the level of the neurons: in every step at most one rule is used in each of them. In a generating system, one of the neurons is the designated *output neuron*, from which spikes are sent to the environment –

eventually only the difference between the first two spikes is considered to be the output number, see [7] – or else are collected as output, whereas in an analyzing (or accepting) system (*spiking neural P automaton*) the designated *input neuron* contains the initial number to be analyzed (accepted) as the number of spikes in the cell or else we take the difference between the first two input spikes as input.

Going back to neural biology, it is worth mentioning that the effect of a spike on the postsynaptic neuron can be measured, and is called the *membrane potential* (the potential difference between the interior of the cell and its surroundings). If the neuron has no electrical activity, its membrane potential is constant. After the arrival of a single spike, the potential changes and finally *decays* back to the resting potential. Hence, if no other spikes arrive in a certain amount of time in the postsynaptic neuron, the initial spike is lost (disappears) and has no further effect on the neuron.

In this paper we incorporate this phenomenon in SNP systems. We also model the *threshold* of the neuron in a different way than it was considered before. More precisely, if the threshold of a neuron is k, then if at any time during the computation the neuron has inside a number of spikes greater or equal to k the neuron must fire, and then erase its whole contents; in a more general way, we here consider *total firing* where a neuron has to empty its whole contents when firing provided its current contents belongs to a given regular set. As the underlying model in which we shall elaborate these ideas of *decaying spikes* and *total firing* we shall take extended spiking neural P systems, see [1], and we shall show that even this extended model does not allow us to go beyond regularity when using *decaying spikes* and/or *total firing.*

2 Preliminaries

For the basic elements of formal language theory needed in the following, we refer to any monograph in this area, in particular, to [3] and [11]. We just list a few notions and notations: V^* is the free monoid generated by the alphabet V under the operation of concatenation and the empty string, denoted by λ, as unit element. $\mathbb{N}_+$ denotes the set of positive integers, $\mathbb{N}$ is the set of non-negative integers (natural numbers), i.e., $\mathbb{N} = \mathbb{N}_+ \cup \{0\}$. For $0 \leq k \leq m$, the interval of natural numbers between k and m is denoted by $[k..m]$. Observe that there is a one-to-one correspondence between a set $N \subseteq \mathbb{N}$ and the one-letter language $L(N) = \{a^n \mid n \in N\}$, hence, N is a regular (semilinear) set of non-negative integers if and only if $L(N)$ is a regular language; on the other hand, for a given one-letter language L the corresponding set of natural numbers $\{n \mid a^n \in L\}$ is denoted by $N(L)$. By FIN and REG we denote the families of finite sets and regular sets of natural numbers, respectively. For a finite set N, $|N|$ denotes the cardinality of N.

A deterministic finite automaton M is a construct (Q, T, δ, q_0, F) where Q is a set of states, T is a set of terminal symbols, $\delta : Q \times T \to Q$ is the transition function, q_0 is the initial state, and $F \subseteq Q$ is a set of final states; in the case of a non-deterministic finite automaton, δ is a finite subset of $Q \times T^* \times Q$. The language accepted by M is denoted by $L(M)$. The regular grammar G corresponding to the non-deterministic finite automaton is $G = (Q, T, P, q_0)$ where P is the set of productions with $P = \{p \to wq \mid (p, w, q) \in \delta\} \cup \{p \to \lambda \mid p \in F\}$. As is well known, the family of languages accepted by (deterministic or non-deterministic) finite automata coincides with the family of regular languages and equals the family of languages generated by regular grammars; hence, the families of one-letter languages accepted by (deterministic or non-deterministic) finite automata or generated by regular grammars coincide with REG.

3 Extended Spiking Neural P Systems

The reader is supposed to be familiar with basic elements of membrane computing, e.g., from [10]; comprehensive information can be found on the P systems web page [12]. Moreover, for the motivation and the biological background of spiking neural P systems we refer the reader to [7]. The following definition is mainly taken from [1]:

An *extended spiking neural P system* (of degree $m \geq 1$) (in the following we shall simply speak of an *ESNP system*) is a construct

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

where

- m is the number of *cells* (or *neurons*); the neurons are uniquely identified by a number between 1 and m (obviously, we could instead use an alphabet with m symbols to identify the neurons);
- S describes the *initial configuration* by assigning an initial value (of spikes) to each neuron; for the sake of simplicity, we assume that at the beginning of a computation we have no pending packages along the axons between the neurons;
- R is a finite set of *rules* of the form $\left(i, E/a^k \to P; d\right)$ such that $i \in [1..m]$ (specifying that this rule is assigned to cell i), $E \subseteq REG$ is the *checking set* (the current number of spikes in the neuron has to be from E if this rule shall be executed), $k \in \mathbb{N}$ is the "number of spikes" (the energy) consumed by this rule, d is the *delay* (the "refraction time" when neuron i performs this rule), and P is a (possibly empty) set of *productions* of the form (l, w, t) where $l \in [1..m]$ (thus specifying the target cell), $w \in \mathbb{N}$ is the *weight* of the energy sent along the axon from neuron i to neuron l, and t is the time needed before the information sent from neuron i arrives at neuron l (i.e., the *delay along the axon*).

A *configuration* of the ESNP system is described as follows:

- for each neuron, the actual number of spikes in the neuron is specified;
- in each neuron i, we may find an "activated rule" $\left(i, E/a^k \to P; d'\right)$ waiting to be executed where d' is the remaining time until the neuron spikes;

- in each axon to a neuron l, we may find pending packages of the form (l, w, t') where t' is the remaining time until w spikes have to be added to neuron l provided it is not closed for input at the time this package arrives.

A *transition* from one configuration to another one now works as follows:

- for each neuron i, we first check whether we find an “activated rule” $\left(i, E/a^k \to P; d'\right)$ waiting to be executed; if $d' = 0$, then neuron i “spikes”, i.e., for every production (l, w, t) occurring in the set P we put the corresponding package (l, w, t) on the axon from neuron i to neuron l, and after that, we eliminate this “activated rule” $\left(i, E/a^k \to P; d'\right)$;
- for each neuron l, we now consider all packages (l, w, t') on axons leading to neuron l; provided the neuron is not closed, i.e., if it does not carry an activated rule $\left(i, E/a^k \to P; d'\right)$ with $d' > 0$, we then sum up all weights w in such packages where $t' = 0$ and add this sum to the corresponding number of spikes in neuron l; in any case, the packages with $t' = 0$ are eliminated from the axons, whereas for all packages with $t' > 0$, we decrement t' by one;
- for each neuron i, we now again check whether we find an “activated rule” $\left(i, E/a^k \to P; d'\right)$ (with $d' > 0$) or not; if we have not found an “activated rule”, we now may apply any rule $\left(i, E/a^k \to P; d\right)$ from R for which the current number of spikes in the neuron is in E and then put a copy of this rule as “activated rule” for this neuron into the description of the current configuration; on the other hand, if there still has been an “activated rule” $\left(i, E/a^k \to P; d'\right)$ in the neuron with $d' > 0$, then we replace d' by $d' - 1$ and keep $\left(i, E/a^k \to P; d' - 1\right)$ as the “activated rule” in neuron i in the description of the configuration for the next step of the computation.

After having executed all the substeps described above in the correct sequence, we obtain the description of the new configuration. A *computation* is a sequence of configurations starting with the initial configuration given by S. A computation is called *successful* if it halts, i.e., if no pending package can be found along any axon, no neuron contains an activated rule, and for no neuron, a rule can be activated.

An ESNP is called *finite* if all the regular checking sets in the rules are finite.

In this paper, however, we will consider the following variants of the above systems:

1. *ESNP systems with decaying spikes:*

   The spikes in the system are decaying, i.e., they only have a limited lifetime before disappearing. In this case, a spike a is now written in the form (a, e), where $e \geq 1$ is the decay that, from the moment a spike (a, e) arrives in a neuron, is decremented by one in each step of the computation. As soon as $e = 0$, the corresponding spike is lost and cannot be used anymore. There could be different strategies with respect to the question which spikes should be consumed when the neuron fires, but these considerations are of no importance for the results elaborated below.

2. *ESNP systems with total spiking:*

   In this case, the whole contents of the neuron is lost as soon as a spiking rule $(i, E/ \to P; d')$ (we omit specifying the number of spikes to be consumed when applying such a rule) can be applied in neuron i as the number of spikes present in the cell is in E.

As a special case of ESNP systems with total spiking we could also consider *ESNP systems with thresholds* where the regular sets E all are of the special form $\{n \in \mathbb{N} \, | n \geq h\}$ with h being the so-called *threshold*. In this case, the rule $(i, E/ \to P; d')$ is also written as $(i, \geq h/ \to P; d')$. We postpone a thorough discussion of these restricted variants for a longer version of this paper, yet we shall use the notation in special examples for ESNP systems with total spiking.

For decaying spikes and total spiking and even a combination of these two, we will consider ESNP systems as generating as well as accepting devices, where the output (input in the case of accepting devices, respectively) is either given in a specified output (input) neuron, or else as the distance between the first two spikes exiting (entering) the system.

4 Results

As throughout this section we do not use delays in the rules and productions, we simply shall omit them to keep the description of the systems concise, e.g., instead of $\left(2, \left\{a^i\right\} / a^i \to \left\{\left(2, a^j, 0\right), (1, a, 0)\right\}; 0\right)$, in the following we shall write $\left(2, \left\{a^i\right\} / a^i \to \left\{\left(2, a^j\right), (1, a)\right\}\right)$.

First we investigate the generative power of extended spiking neural P systems with decaying spikes and/or total spiking; in the following, for generating ESNP systems we shall always assume that the output neuron contains no spiking rules; moreover, the neurons except the output neuron are called *actor neurons*:

When we consider the output to be the number of spikes at the end of a successful computation, then ESNP systems with decaying spikes can only generate finite sets, because the number of spikes that can be added in one step to the contents of a neuron is bounded, but the spikes in a neuron have a limited life-time, hence, at any moment the number of spikes in a neuron is bounded. On the other hand, every finite set of natural numbers can be generated by an extended spiking neural P system with spikes of minimal decay with only two neurons:

Example 1 *Any finite set of natural numbers N can be generated by a finite ESNP system with spikes of minimal decay with only two neurons.*

Let N be a finite set of natural numbers. We now construct the finite ESNP system Π that generates an element of N by the number of spikes contained in the output neuron 1 at the end of a successful computation:

$$\begin{aligned}
\Pi &= (2, S, R), \\
S &= \{(1, \lambda), (2, (a, 1))\}, \\
R &= \left\{\left(2, \{(a, 1)\} / (a, 1) \to \left\{\left(1, (a, 1)^j\right)\right\}\right) \mid j \in N\right\};
\end{aligned}$$

after one step, every computation halts, the output neuron having received a number of spikes corresponding to a number from N. We could even add the feature of total spiking or minimal threshold 1 in order to obtain the same result; in this case, R would be written as

$$\left\{\left(2,\{(a,1)\}/\rightarrow\left\{\left(1,(a,1)^j\right)\right\}\right)\mid j\in N\right\}$$

or

$$\left\{\left(2,\geq 1/\rightarrow\left\{\left(1,(a,1)^j\right)\right\}\right)\mid j\in N\right\}.$$

For the sake of completeness, we should like to mention that the empty set is generated by the ESNP system

$$\begin{array}{rcl}\Pi_0 & = & (2,S,R_0)\,,\\ S & = & \{(1,\lambda)\,,(2,(a,1))\}\,,\\ R_0 & = & \{(2,\geq 1/\rightarrow\{(2,(a,1))\})\}\,,\end{array}$$

which only has one infinite computation.

In sum, the ESNP systems with decaying spikes (even together with total spiking) generating the result as the number of spikes in the output neuron at the end of a successful computation characterize the finite sets of natural numbers:

Theorem 2 *Any finite set of natural numbers N can be generated by a ESNP system with spikes of minimal decay with only two neurons (even with total spiking, too). On the other hand, every set of natural numbers generated in the output neuron by an ESNP system with decaying spikes (even with total spiking, too) is finite.*

If we only consider the output to be the difference between the first two spikes arriving in the output neuron during a halting computation, then we obtain a characterization of the regular sets of natural numbers even with ESNP systems with decaying spikes:

Example 3 *Let N be a regular set of natural numbers accepted by the deterministic finite automaton $M=(Q,\{a\},\delta,1,F)$ with $Q=[1..m]$. Then $L(M)$ is generated as the difference between the (first) two spikes arriving in the output neuron by the following ESNP system with decaying spikes and total spiking Π' with the output neuron 1:*

$$\begin{array}{rcl}\Pi' & = & (2,S',R')\,,\\ S' & = & \left\{(1,\lambda)\,,\left(2,(a,1)^{m+1}\right)\right\},\\ R' & = & \left\{\left(2,\left\{(a,1)^i\right\}/\rightarrow\left\{\left(2,(a,1)^j\right)\right\}\right)\mid i,j\in[1..m]\,,\delta(i,a)=j\right\}\\ & \cup & \left\{\left(2,\left\{(a,1)^i\right\}/\rightarrow\{(1,(a,1))\}\right)\mid i\in[1..m]\,,i\in F\right\}\\ & \cup & \left\{\left(2,\left\{(a,1)^{m+1}\right\}/\rightarrow\{(1,(a,1))\,,(2,(a,1))\}\right)\right\}.\end{array}$$

Obviously, this system can also be interpreted as ESNP with having only one of the features decaying spikes and total spiking.

If only the restricted variant of total spiking with thresholds is used, then we need a more complicated ESNP system Π'' *(without or even with decaying spikes) where each state* i *of* M *is represented by the neuron* $i+1$*:*

$$\begin{array}{rcl}
\Pi'' & = & (m+2, S'', R''), \\
S'' & = & \{(m+2,(a,1))\} \cup \{(i,\lambda) \mid i \in [1..m+1]\}, \\
R'' & = & \{(i, \geq 1/ \rightarrow \{(j,(a,1))\}) \\
 & & \mid i,j \in [2..m+1], \delta(i-1,a) = j-1\} \\
 & \cup & \{(i, \geq 1/ \rightarrow \{(1,(a,1))\}) \mid i \in [2..m+1], i-1 \in F\} \\
 & \cup & \{(m+2, \geq 1/ \rightarrow \{(1,(a,1)), (2,(a,1))\})\}.
\end{array}$$

In fact, the control set $\{n \geq 1\}$ *could be replaced by the finite set* $\{1\}$*, i.e.,* Π'' *corresponds to a finite system.*

A slight modification of the ESNP system Π'' *yields the ESNP system* Π''' *which generates* $L(M)$ *as the number of spikes in the output neuron even with the minimal threshold, but obviously only without decays:*

$$\begin{array}{rcl}
\Pi''' & = & (m+1, S'', R'''), \\
S'' & = & \{(2,a)\} \cup \{(i,\lambda) \mid i \in \{1\} \cup [3..m+1]\}, \\
R''' & = & \{(i, \geq 1/ \rightarrow \{(1,a), (j,a)\}) \\
 & & \mid i,j \in [2..m+1], \delta(i-1,a) = j-1\} \\
 & \cup & \{(i, \geq 1/ \rightarrow \emptyset) \mid i \in [2..m+1], i-1 \in F\}.
\end{array}$$

In [1] it was shown that every ESNP system where the number of spikes remains bounded can only generate regular sets. The same arguments used to prove this result immediately show that ESNP systems with decaying spikes can only generate regular sets because the number of spikes is bounded in these cases and therefore the behaviour of the ESNP systems can be modeled by a (non-deterministic) finite automaton. Yet the same also holds true for ESNP systems with total firing:

Theorem 4 *Every language generated by an ESNP system with total firing is regular.*

Proof (sketch). Let Π be an ESNP system with total firing. Then the regular sets used in the rules of Π are of a very special form, i.e., they are a finite union of very simple sets which either are equal to $\{y\}$ or of the form $\{xn+y \mid n \in \mathbb{N}\}$ with $x, y \in \mathbb{N}$ and $x \neq 0$. Hence, it is sufficient to store the actual contents of a neuron as a vector remembering for each of these sets either the value until it exceeds y for $\{y\}$ and the module class after exceeding y for $\{xn+y \mid n \in \mathbb{N}\}$, i.e., in sum we only have a finite number of possible states of each neuron we have to consider instead of the actual values which eventually might go beyond any fixed bound.

Then the number of configurations differing in the actor neurons (i.e., the neurons except the output neuron) and the packages along the axons, but without considering the contents of the output neurons, is finite, hence, we can assign non-terminal symbols A_k to each of these configurations and take all right-regular productions $A_i \rightarrow a^k A_j$ such that k is the number of spikes added the output neuron when going from configuration i to configuration j. The initial configurations is the start symbol of the regular grammar G constructed in that way, and, finally, for all halting configurations i we add the production $A_i \rightarrow \lambda$. In that way we can construct a regular grammar generating a one-letter language corresponding with the set of natural number generated by Π in the output neuron.

These considerations can also be taken over to the case when the output is taken as the difference between the (first) two spikes arriving in the output neuron: here we use productions of the form $A_i \rightarrow A_j$ for the periods before the first spike appears in the output neuron; afterwards we take productions $A_i \rightarrow aA_j$, i.e., for each time step in Π we generate one symbol a in a derivation in G. After the second spike has appeared in the output neuron we continue again with productions of the form $A_i \rightarrow A_j$, and as for the previous case we finish with productions $A_i \rightarrow \lambda$ for all halting configurations i. $\square$

Hence, we can summarize these results characterizing REG for the generating cases as follows:

Theorem 5 *Any regular set $N \in REG$ can be generated by an ESNP system with decaying spikes and/or total firing and the output taken as the difference between the first two spikes arriving in the output neuron during a successful computation; moreover, N can also be generated by an ESNP system with total firing and the output given as the number of spikes in the output neuron at the end of a successful computation. On the other hand, every language generated by an ESNP system with total firing or by an ESNP system with decaying spikes and/or total firing and the output being the difference between the first two spikes arriving in the output neuron during a successful computation is regular.*

Now we consider the *accepting case* where the case of the input being given as the number of spikes in the input neuron (we always assume that the input neuron gets its input only from the environment) is quite trivial:

Example 6 *Let N be a regular set of natural numbers. Then N is accepted by the ESNP system with decaying spikes and/or total firing*

$$\begin{array}{rcl} \Pi(N) & = & (2, \{(1,\lambda), (2,\lambda)\}, R(N)), \\ R(N) & = & \{(1, L(\mathbb{N} - N) / \rightarrow \{(2,(a,1))\}), (2, \{a\} / \rightarrow \{(2,(a,1))\})\}. \end{array}$$

The input n is given by $(a,1)^n$ in the first neuron which fires if and only if $n \notin N$, hence the infinite loop in the second neuron is only started in this case, whereas for $n \in N$ the system immediately halts. We should like to mention that the spike consumed by the rule $(2, \{a\} / \rightarrow \{(2,(a,1))\})$ will always be a decaying spike $(a,1)$.

Example 7 *Let N be a regular set of natural numbers accepted by the deterministic finite automaton $M = (Q, \{a\}, \delta, 1, F)$ with $Q = [1..m]$. Then $N(L(M))$ is accepted as the input being given as the difference between the first and the second spike arriving in the input neuron by the following finite ESNP system with total spiking* *even when using spikes with minimal decay:*

$$\begin{array}{rcl}
\Pi^t &=& \left(m+5, S^t, R^t\right), \\
S^t &=& \{(2,(a,1))\} \cup \{(i,\lambda) \mid i \in \{1\} \cup [3..m+5]\}, \\
R^t &=& \{(1, \{(a,1)\} / \to \{(m+2,(a,1)), (m+3,(a,1)), (m+4,(a,1))\})\} \\
&\cup& \{(m+2, \{(a,1)\} / \to \{(m+2,(a,1)), (m+4,(a,1))\})\} \\
&\cup& \left\{\left(m+2, \left\{(a,1)^2\right\} / \to \left\{\left(m+3,(a,1)^2\right), (m+5,(a,1))\right\}\right)\right\} \\
&\cup& \{(m+3, \{(a,1)\} / \to \emptyset)\} \\
&\cup& \left\{\left(m+3, \left\{(a,1)^2\right\} / \to \left\{\left(m+3,(a,1)^2\right)\right\}\right)\right\} \\
&\cup& \left\{\left(m+3, \left\{(a,1)^3\right\} / \to \{(j,(a,1)) \mid j \in [2..m+1]\}\right)\right\} \\
&\cup& \{(m+4, \{(a,1)\} / \to \emptyset)\} \\
&\cup& \left\{\left(m+4, \left\{(a,1)^2\right\} / \to \{(m+5,(a,1))\}\right)\right\} \\
&\cup& \{(m+5, \{(a,1)\} / \to \emptyset)\} \\
&\cup& \left\{\left(m+5, \left\{(a,1)^2\right\} / \to \left\{\left(2,(a,1)^2\right)\right\}\right)\right\} \\
&\cup& \left\{\left(i, \left\{(a,1)^2\right\} / \to \left\{\left(j,(a,1)^2\right)\right\}\right)\right. \\
&& \left.\mid i,j \in [2..m+1], \delta(i-1,a) = j-1\right\} \\
&\cup& \left\{\left(i, \left\{(a,1)^3\right\} / \to \emptyset\right) \mid i \in [2..m+1], i-1 \in F\right\}.
\end{array}$$

The neuron $m+2$ keeps the computation alive until the input neuron 1 spikes for the first time. This starting impulse is also propagated through neurons $m+4$ and $m+5$ to neuron 2 (which corresponds to the initial state of M). This delay through neurons $m+4$ and $m+5$ allows for starting the loop in neuron $m+3$ in time, because this loop can only be ceased by the second spike arriving in the input neuron, which then also has to be propagated to the neuron i representing the actual state $i-1$ of M and to halt the computation in Π^t by applying the rule $\left(i, \left\{(a,1)^3\right\} / \to \emptyset\right)$ provided $i-1$ is a final state in M.

Again similar arguments as in Theorem 5 can be applied when considering ESNP systems accepting a number given as this number of spikes in the input neuron or else as the difference between the (first) two spikes introduced in the input neuron from the environment:

Theorem 8 *Every set of natural numbers accepted by an ESNP system with total firing and/or decaying spikes is regular.*

Proof (sketch). Let Π be an ESNP system with total firing and/or decaying spikes. We then construct a (non-deterministic) finite automaton M accepting the regular language corresponding with the set of natural numbers accepted by Π:

First we consider the case where the input is given as the number of spikes in the input neuron. For an ESNP system with total spiking, we first construct a finite automaton M' which analyses the given input according to the rules of Π in the input neuron, which, as already elaborated in the proof of Theorem 5 are of a very special form, i.e., they are a finite union of very simple sets which either are equal to some singleton set $\{y\}$ or of the form $\{xn+y \mid n \in \mathbb{N}\}$ with $x, y \in \mathbb{N}$ and $x \neq 0$.

Hence, it is sufficient to evaluate the contents of the input neuron to a state of M' which represents a vector remembering for each of these sets either the value until it exceeds y for $\{y\}$ and the module class after exceeding y for $\{xn + y \mid n \in \mathbb{N}\}$. According to the state of M' finally reached with the input string, we then know whether the input neuron would spike or not. M then consists of M' and M'' where M'' is constructed to start with the information from the computation in M' and then simulates the transitions in Π without taking into account the input neuron anymore. M'' only makes λ-transitions until it reaches a state corresponding to a halting configuration; exactly these states corresponding to a halting configuration are the final states of M, i.e., M halts in a final state if and only if Π halts. If we add the feature of decaying spikes, this has no influence on M', we only have to take it into account for M''. Observe that in any case, the construction of M'' again relies on the finiteness of the description of the actual contents of the neurons possible for total firing. On the other hand, if we have only decaying spikes (a, e) given in the input neuron, but no total spiking, after e steps no further information is left in the input neuron. Hence, we can apply a similar construction for a finite automaton M where we integrate the possible states of the input neuron in the possible behaviour of the whole system which remains bounded due to the fact that with decaying spikes the maximal number of spikes in the whole system remains bounded after the first e steps.

If the input is given as the difference between the first two spikes in the input neuron, then we have to construct M in three substeps: In the first step, the ESNP system works without taking into consideration the input cell. In all cases, i.e., working with decaying spikes and/or total spiking, we only get a finite set of possible states Q_I and corresponding transitions between them which exactly simulate the behaviour of the ESNP system. For every state $q \in Q_I$ we now take a state q' which only differs from q by having one spike in the input neuron; in that way we get a set Q'_I describing the configurations of Π when the first input spike arrives. Starting with Q'_I we now compute all possible configurations and the transitions between them, which yields the set Q_C. For every state $q \in Q_C$ we now take a state q' which only differs from q by having one spike in the input neuron; in that way we get a set of states Q'_C describing the configurations of Π when the second input spike arrives. The states in Q'_C are the starting point for the third and last step of the simulation, which again can be described by a finite set of states Q_O and the transitions between them. For getting M from Q_I, Q'_I, Q_C, Q'_C, and Q_O and the transitions between these states, we take the union of all these states as the set of states for M; the initial state is the state from Q_I corresponding to the initial configuration; the final states are those states from Q_O that correspond to a halting configuration; the transitions between the states in Q_I, between the states in Q_I and those in Q'_I, the transitions between the states in Q_C and those in Q'_C, as well as the transitions between the states in Q_O are λ-transitions in M, whereas a transition between the states p and q in Q_C corresponds with an a-transition (p, a, q) in M, i.e., each time step between the first and the second spike arriving in the input neuron in Π consumes one symbol a in M. This observation completes the proof. $\square$

Hence, we can summarize the results characterizing REG for the accepting cases as follows:

Theorem 9 *Any regular set $N \in REG$ can be accepted by an ESNP system with decaying spikes and/or total firing, the input either being given in the input neuron or else being taken as the difference between the first two spikes arriving in the input neuron during a successful computation. On the other hand, every language accepted by an ESNP system with decaying spikes and/or total firing, the input either being given in the input neuron or else being taken as the difference between the first two spikes arriving in the input neuron from the environment during a successful computation, is regular.*

5 Conclusion

In this paper, we have investigated extended spiking neural P systems with decaying spikes and/or total spiking, and we have proved that even when combining decaying spikes and/or total spiking we get a characterization of the regular sets of natural numbers with these systems being considered as generating devices or else as accepting devices (automata), except for the following cases: extended spiking neural P systems with decaying spikes (even with total spiking, too) used as generating devices with the output being given as the number of spikes in the output neuron at the end of a successful computation yield a characterization of the finite sets.

In an extended version we shall also investigate the generating and accepting power of spiking neural P systems incorporating only the original features or even more restricted variants (e.g., see [6]) as well as decaying spikes and/or total spiking. Moreover, ESNP systems with thresholds deserve further investigations. Finally, (E)SNP systems should also be considered as generators or acceptors for sets of vectors of natural numbers as well as even of string languages (e.g., compare [2]).

6 Acknowledgements

The work of Marion Oswald is supported by FWF-project T225-N04.

References

[1] A. Alhazov, R. Freund, M. Oswald, M. Slavkovik. Extended Spiking Neural P Systems Generating Strings and Vectors of Non-Negative Integers. In: H.J. Hoogeboom, Gh. Păun, G. Rozenberg, editors, *Workshop on Membrane Computing, WMC7, Leiden, The Netherlands 2006*, LNCS 4361, pages 123-134. Springer, 2007.

[2] H. Chen, R. Freund, M. Ionescu, Gh. Păun, M. J. Pérez-Jiménez. On String Languages Generated by Spiking Neural P Systems. In M.A. Gutiérrez-Naranjo, Gh. Păun, A. Riscos-Núñez, F. José Romero-Campero, editors, *Fourth Brainstorming Week on Membrane Computing, Vol. I,* RGNC REPORT 02/2006, pages 169-194. Research Group on Natural Computing, Sevilla University, Fénix Editora, Sevilla, 2006.

[3] J. Dassow, Gh. Păun. *Regulated Rewriting in Formal Language Theory.* Springer-Verlag, Berlin, 1989.

[4] R. Freund, Gh. Păun, M.J. Pérez-Jiménez. Tissue-like P systems with channel states. *Theoretical Computer Science*, 330:101–116, 2005.

[5] W. Gerstner, W. Kistler. *Spiking Neuron Models. Single Neurons, Populations, Plasticity.* Cambridge Univ. Press, 2002.

[6] O. H. Ibarra, A. Păun, Gh. Păun, A. Rodríguez-Patón, P. Sosík, S. Woodworth. Normal Forms for Spiking Neural P Systems. In M. A. Gutiérrez-Naranjo, Gh. Păun, A. Riscos-Núñez, F. José Romero-Campero, editors, *Fourth Brainstorming Week on Membrane Computing, Vol. II*, RGNC REPORT 02/2006, pages 105-136. Research Group on Natural Computing, Sevilla University, Fénix Editora, Sevilla, 2006.

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

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

[9] W. Maass, C. Bishop, editors. *Pulsed Neural Networks.* MIT Press, Cambridge, 1999.

[10] Gh. Păun. *Membrane Computing: An Introduction.* Springer-Verlag, Berlin, 2002.

[11] G. Rozenberg, A. Salomaa, editors. *Handbook of Formal Languages* (3 volumes). Springer-Verlag, Berlin, 1997.

[12] The P Systems Web Page, `http://psystems.disco.unimib.it`