Characterizations of Some Restricted Spiking Neural P Systems ⋆ (Extended Abstract)

Oscar H. Ibarra and Sara Woodworth
Department of Computer Science
University of California, Santa Barbara, CA 93106, USA

Abstract. A k-output spiking neural P system (SNP) with output neurons, $O_1, ..., O_k$, generates a tuple $(n_1, ..., n_k)$ of positive integers if, starting from the initial configuration, there is a sequence of steps such that during the computation, each O_i generates exactly two spikes a a (the times the pair a a are generated may be different for different output neurons) and the time interval between the first a and the second a is n_i. After the output neurons have generated their pairs of spikes, the system eventually halts. Another model, called k-train SNP, has only one output neuron. It generates a k-tuple $(n_1, ..., n_k)$ if, starting from the initial configuration, the output neuron O generates the spike train $aa...a$ with exactly $k+1$ a's such that the interval between the i^{th} a and the $i+1^{st}$ a is n_i, and the system eventually halts. We assume, without loss of generality, that each neuron in the SNP is either bounded or unbounded. (Bounded here means that there is a fixed constant c such that at any time during the computation, the number of spikes in the neuron is at most c. Otherwise, the neuron is unbounded.) It is known that 1-output SNPs (= 1-train SNPs) are universal, i.e., they generate exactly the recursively enumerable sets over N. Here, we show the following:

1. For $k \geq 1$, a set $Q \subseteq N^k$ is semilinear if and only if it can be generated by a k-output SNP, where every unbounded neuron satisfies the property that once it starts "spiking" it will no longer receive future spikes (but can continue spiking). This result also holds for k-train SNP.
2. The set $Q = \{(m, 2m) \mid m \geq 1\}$ (which is semilinear) cannot be generated by any 2-output bounded SNP (i.e., SNP all of whose neurons are bounded). Thus, for $k \geq 2$, there are semilinear sets over N^k that cannot be generated by k-output bounded SNPs. This contrasts a known result that 1-output bounded SNPs generate all semilinear sets over N.
3. For $k \geq 2$, k-output bounded SNPs are computationally more powerful than k-train bounded SNPs. (They are identical when $k = 1$.)
4. For $k \geq 1$, k-output bounded SNPs and k-train bounded SNPs can be characterized by certain classes of nondeterministic finite automata with monotonic counters.

Keywords: Spiking neural P system, k-output SNP, k-train SNP, bounded neuron, unbounded neuron, semilinear set, counter machine.

⋆ This research was supported in part by NSF Grants CCF-0430945 and CCF-0524126.

1 Introduction

Spiking neural P systems (SNPs) were recently introduced in [6], and investigated in a series of papers: [11], [12], [5]. The model of an SNP incorporates into membrane computing [10] ideas from spiking neurons, see, e.g., [1], [7], [8].

An SNP consists of a set of neurons placed in the nodes of a graph. The neurons send signals (spikes) along synapses (directed edges of the graph). This is done by means of firing rules, which are of the form $E/a^c \to a; t$, where E is a regular expression, c is the number of spikes consumed by the rule, and t is the delay from firing the rule and emitting the spike. The rule can be used only if the number of spikes in the neuron is "covered" by expression E, in the sense that the current number of spikes in the neuron, n, is such that a^n is contained in the set $L(E)$ denoted by the expression E. In the time interval between firing a rule and emitting the spike, the neuron is closed/blocked – it does not receive other spikes and cannot fire. After the time interval, the neuron is again open and can again fire and receive other spikes. There also are rules for forgetting spikes, of the form $a^s \to \lambda$ (s spikes are just removed from the neuron). We require that if $a^s \to \lambda$ is a rule in a neuron, then a^s is not in E for any firing rule of the form $E/a^c \to a; t$ in the neuron. (Thus, forgetting rules and the firing rules must be disjoint within each neuron.) Starting from a fixed initial distribution of spikes in the neurons (initial configuration) and using the rules in a synchronized manner (a global clock is assumed), the system evolves. A computation is a sequence of transitions starting from the initial configuration. A transition is maximally parallel in the sense that all neurons that are fireable must fire. However, in any neuron, at most one rule is allowed to fire. Details can be found in [6].

An SNP can be used as a computing device in various ways. Here, as in previous papers, we will use them as generators of numbers. We will only consider SNPs with two types of neurons:

1. A neuron is *bounded* if every rule in the neuron is of the form $a^i/a^j \to a; t$, where $j \leq i$, or of the form $a^k \to \lambda$, provided there is no rule of the form $a^k/a^j \to a; t$ in the neuron. Note that there can be several such rules in the neuron. These rules are called *bounded rules*. (For notational convenience, we will write $a^i/a^i \to a; t$ simply as $a^i \to a; t$.)
2. A neuron is *unbounded* if every rule in the neuron is of the form $E/a^c \to a; t$, where E denotes an infinite (unary) regular set. Examples of such rules are: $a^3(a^2)^*/a^2 \to a; t$, $a^3(a^2)^*/a^3 \to a; t$, $a(a)^*/a \to a; t$. (Again, there can be several such rules in the neuron.) These rules are called *unbounded rules*.

An SNP is bounded if all the neurons in the system are bounded. If, in addition, there are unbounded neurons then the SNP is said to be unbounded.

We generalize the SNP by allowing it to produce k outputs. A *k-output SNP Π* has k output neurons, $O_1, ..., O_k$. We say that Π generates a k-tuple $(n_1, ..., n_k) \in N^k$ if, starting from the initial configuration, there is a sequence of steps such that each output neuron O_i generates exactly two spikes a a (the times the pairs a a are generated may be different for different output neurons)

and the time interval between the first a and the second a is n_i. Moreover, after all the output neurons have generated their pair of spikes, the system eventually *halts* in the sense that it reaches a configuration where all neurons are open but no neurons are fireable. The set of all k-tuples generated is denoted by $Q(\Pi)$.

It was recently shown in [5] that a set $Q(\Pi) \subseteq N^1$ is recursively enumerable if and only if it can be generated by a 1-output unbounded SNP Π all of whose unbounded neurons have only one rule, and it is either $a(a)^*/a \rightarrow a; 0$ or $a(a)^*/a \rightarrow a; 2$.

It was also shown in [6] that semilinear subsets of N^1 can be characterized by bounded SNPs. However, it turns out that bounded k-output SNPs cannot generate all semilinear subsets of N^k, when $k \geq 2$. For example, we show in this paper that the set of tuples $\{(n, 2n) \mid n \geq 1\}$ cannot be generated by a 2-output SNP. We then give a characterization of subsets of N^k generated by k-output bounded SNPs.

For semilinear subsets of N^k (for any k), we show that they can be characterized by k-output unbounded SNPs where every unbounded neuron satisfies the property that that once it starts "spiking" it will no longer receive future spikes (but can continue spiking). Such SNPs are called 1-reversal-bounded.

We then look at another restricted model called *k-train SNP*. Such a system has only one output neuron. Again, there are two types of neurons: bounded and unbounded but 1-reversal-bounded, as defined before. We say that Π generates the k-tuple $(n_1, ..., n_k)$ if, starting from the initial configuration, the output neuron O generates the spike train $aa...a$ with exactly $k+1$ outputted a's such that the interval between the i^{th} a and the $i+1^{st}$ a is n_i, and the system eventually halts. We show that 1 reversal-bounded k-train unbounded SNPs also characterize the semilinear sets over N^k. We also give a characterization of k-train bounded SNPs and show that they cannot generate all semilinear sets. In fact, they are weaker than k-output bounded SNPs.

We note that in this extended abstract, all results with the exception of Theorem 1, are stated without proofs. The proofs will appear in the full paper.

2 Characterization of k-Output Bounded SNPs

We first recall the definition of a semilinear set. A set $Q \subseteq N^k$ is a *linear set* if there exist vectors $v_0, v_1, \ldots, v_t$ in N^k such that

$$Q = \{v \mid v = v_0 + m_1 v_1 + \cdots + m_t v_t,\ m_i \in N\}.$$

The vectors v_0 (referred to as the *constant vector*) and $v_1, v_2, \ldots, v_t$ (referred to as the *periods*) are called the *generators* of the linear set Q. A set $Q \subseteq N^k$ is *semilinear* if it is a finite union of linear sets. The empty set is a trivial (semi)linear set, where the set of generators is empty. Every finite subset of N^k is semilinear – it is a finite union of linear sets whose generators are constant vectors. It is also clear that the semilinear sets are closed under (finite) union. It is also known that they are closed under union, complementation, intersection,

and projection. A semilinear subset of N^1 (i.e., 1-tuples) is sometimes referred to as regular.

It is known that 1-output SNPs with only bounded neurons generate exactly the semilinear sets over N^1 [6]. However, as we shall see in Theorem 1, when $k \geq 2$, k-output SNPs with only bounded neurons cannot generate all semilinear sets over N^k. But these SNPs can generate some simple semilinear sets. For example Figure 1 is an SNP with only bounded neurons generating the set $\{(m, n, m+n) \mid m, n \geq 2\}$.

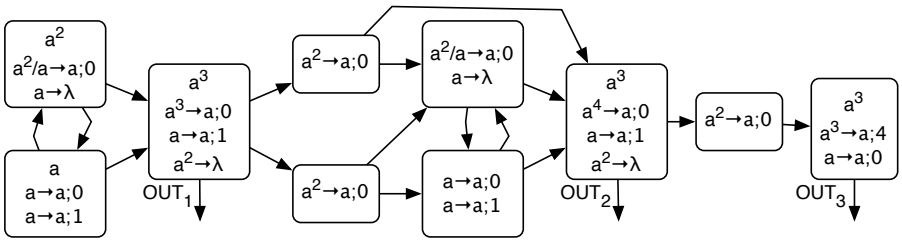

Fig. 1. Bounded SNP Generating $Q = \{(m, n, m+n) \mid m, n \geq 2\}$.

Theorem 1. *The set $Q = \{(m, 2m) \mid m \geq 1\}$ cannot be generated by any 2-output bounded SNP.*

Proof. Let Π be an SNP with m neurons that are all bounded. The distribution of the spikes in the neurons and states of the neurons corresponding to the spiking intervals specified by the last rules used in each neuron (the open-close status and the time since the neurons were closed, depending on the rule used) and the rule to be used next define the configuration of the system. It is important to note that our definition of configuration includes the next rule that can be applied in the neuron. Clearly there are at most n configurations that the system can be in, for some n. Let O_1 and O_2 be the output neurons.

Suppose Π generates the set $Q = \{(m, 2m) \mid m \geq 1\}$. Let $s = n+1$. Then Π generates $(s, 2s)$. Let t_1 and t_2 be the times the output membrane O_1 generates the first and second spikes, respectively. Hence, $t_2 - t_1 = s$. Similarly, let t_3 and t_4 be the times O_2 spikes; hence $t_4 - t_3 = 2s$. We consider two cases:

Case 1. Suppose $t_1 \leq t_3$. Then we claim that $t_3 - t_1 \leq n$. Otherwise, in the time interval between t_1 and t_3, the system will enter some configuration C twice at times t'_1 and t'_3, where $t_1 \leq t'_1 < t'_3 \leq t_3$. Let $t'_3 - t'_1 = r$. (Note that $r \leq n$.) Then, clearly the tuple $(s + kr, 2s)$ is also generated by Π for each $k \geq 1$. This is a contradiction.

Case 2. Now suppose $t_3 \leq t_1$. Then, again, $t_1 - t_3 \leq n$; otherwise (by an argument similar to Case 1), for some $r \leq n$, $(s, 2s + kr)$ will also be generated by Π for each $k \geq 1$, a contradiction.

Thus we may assume that $|t_3 - t_1| \leq n$. But this would imply that $t_4 - t_2 \geq n$. Then for some $r \leq n$, $(s, 2s + kr)$ will also be generated by Π for each $k \geq 1$, which is again a contradiction. □

Next, we give a characterization of subsets of N^k generated by k-output bounded SNPs. Consider a nondeterministic finite automaton $\mathcal{M}$ with k counters, all of which are output counters.

1. Initially all counters are zero.
2. No counter is decremented during the computation.
3. In one step, zero or more counters can be incremented (by 1).
4. When a counter is incremented, it gets incremented at every step until such time when it stops incrementing. Thereafter, the counter no longer increments.
5. Each counter gets incremented at some point.
6. When all the counters have stopped incrementing, the machine may continue computing but eventually halts in an accepting state. The values in the k counters is then said to be generated by $\mathcal{M}$.

Note that item 4 is an important restriction on the operation of the machine. Call the counter machine (CM) described above a *monotonic k-output CM*. Clearly, the set $\{(m, n, m + n) \mid m, n \geq 1\}$ can be generated by a monotonic 3-output CM. Other examples are: $\{(n, n) \mid n \geq 1\}$, $\{(k, n) \mid k, n \geq 1, k < n\}$, and $\{(m, k, n,) \mid m, k, n \geq 1, m < k < n\}$. The first two can be generated by monotonic 2-output CMs, and the last can be generated by a monotonic 3-output CM.

Formally we can define a monotonic k-output CM as $\mathcal{M} =< k, B, l_1, l_h, R >$ where k is the number of counters in the system, B is the set of instruction labels, l_1 is the starting instruction, l_h is the halting instruction, and R is the set of instructions. The instructions in R are of the form

$l_i : (\texttt{BEGIN}(r_1, \cdots, r_p), \texttt{END}(s_1, \cdots, s_q), l_{i_1}, \cdots, l_{i_t})$
$l_i : (\texttt{DELAY}, l_{i_1}, \cdots, l_{i_t})$
$l_i : (\texttt{HALT})$

The counters are indexed $1, ..., k$. The instruction $l_i : (\texttt{BEGIN}(r_1, \cdots, r_p), \texttt{END}(s_1, \cdots, s_q), l_{i_1}, \cdots, l_{i_t})$ starts incrementing counters $r_1, ..., r_p$ (this assumes these counters were zero) and stops incrementing counters $s_1, ..., s_q$ (this assumes these counters have been incrementing at every step, since they started incrementing earlier). The set $\{r_1, ..., r_p\}$ (respectively, $\{s_1, ..., s_q\}$) is called the incrementing set (respectively, end-incrementing set) of the instruction. The counters in the incrementing set begin incrementing during the current step, but the counters in the end-incrementing step are stopped before they are incremented during the current step. If $p = 0$ or $q = 0$, we do not write BEGIN or END. In particular, if $p = q = 0$, then the instruction reduces to $l_i : (l_{i_1}, ..., l_{i_t})$, which

we denote by the instruction $l_i : (\texttt{DELAY}, l_{i_1}, ..., l_{i_t})$ for clarity. During this instruction no counter changes state but all previously incrementing counters are incremented by one and then the system changes state. We emphasize that once a counter r has begun incrementing, it is incremented at each proceeding step until r is in the end-incrementing set of the current instruction. If r is ever incremented again, the computation is invalid. The incrementing set must be disjoint of the end-incrementing set (since we cannot begin and end a counter in a single step). The instruction $l_i : (\texttt{HALT})$ halts the execution. We can assume without loss of generality that no instruction loops back to itself. Note that if $l_{i_1}, \cdots, l_{i_t}$ are identical (i.e., the next instruction is unique), we need only put one such label.

To better understand how monotonic CMs operate we give the following examples.

Example 1. The set $Q = \{(n, n) \mid n \geq 1\}$ can be generated by the monotonic CM $\mathcal{M} =< 2, \{l_1, \cdots, l_5\}, l_1, l_5, R >$ where

$$\begin{aligned} R = \{\ & l_1 : (\texttt{BEGIN}(1,2), l_2, l_4), \\ & l_2 : (\texttt{DELAY}, l_3, l_4), \\ & l_3 : (\texttt{DELAY}, l_2, l_4), \\ & l_4 : (\texttt{END}(1,2), l_5), \\ & l_5 : (\texttt{HALT})\} \end{aligned}$$

This program begins incrementing the counters at time 1. The (possibly executed) delay instructions allow some nondeterministic amount of time to elapse. Then we end the incrementing of both counters at time $n+1$. This gives us the tuple (n, n) where $n \geq 1$. (Note that the counters increment during the time step they are begun guaranteeing that each counter contains an output which is ≥ 1.)

Example 2. The set $Q = \{(m, k, n) \mid m \leq k \leq n, m \geq 1\}$ can be generated by the monotonic CM $\mathcal{M} =< 3, \{l_1, \cdots l_{14}\}, l_1, l_{14}, R >$ where

$$\begin{aligned} R = \{\ & l_1 : (\texttt{BEGIN}(1,2,3), l_2, l_4, l_5, l_6), \\ & l_2 : (\texttt{DELAY}, l_3, l_4.l_5, l_6), \\ & l_3 : (\texttt{DELAY}, l_2, l_4, l_5, l_6), \\ & l_4 : (\texttt{END}(1), l_7, l_9, l_{10}), \\ & l_5 : (\texttt{END}(1,2), l_{11}, l_{13}), \\ & l_6 : (\texttt{END}(1,2,3), l_{14}), \\ & l_7 : (\texttt{DELAY}, l_8, l_9, l_{10}), \\ & l_8 : (\texttt{DELAY}, l_7, l_9, l_{10}), \\ & l_9 : (\texttt{END}(2), l_{11}, l_{13}), \\ & l_{10} : (\texttt{END}(2,3), l_{14}), \\ & l_{11} : (\texttt{DELAY}, l_{12}, l_{13}), \\ & l_{12} : (\texttt{DELAY}, l_{11}, l_{13}), \\ & l_{13} : (\texttt{END}(3), l_{14}), \\ & l_{14} : (\texttt{HALT})\} \end{aligned}$$

The following characterizes k-output bounded SNPs.

Theorem 2. *A set $Q \subseteq N^k$ is generated by a k-output bounded SNP if and only if Q is generated by a monotonic k-ouput CM.*

The characterization allows us to easily show that a set is generated by a k-output bounded SNP by simply showing that it can be generated by a monotonic k-output CM. So, e.g., the sets in Examples 1 and 2 can be generated by 2-output and 3-output bounded SNPs, respectively.

Suppose we relax the operation of a monotonic k-output CM so that we no longer require that when a counter is incremented, it gets incremented at every step until such time when it stops incrementing. Thus, each counter need not be incremented at each step, but eventually, the machine halts in an accepting state when each counter has value at least 1. This type of machine (called 0-reversal CM in the next section) is more powerful than a monotonic k-output CM, since such CMs generate exactly the semilinear sets. In the next section, we characterize them in terms of restricted k-output unbounded SNPs.

3 Reversal-Bounded k-Output Unbounded SNPs

In order to characterize all the semilinear sets over N^k, we need unbounded neurons that operate in a restricted manner. A k-output SNP is *reversal-bounded* if there is an integer $r \geq 0$ such that for each unbounded neuron, the number of times the spike size changes values from nonincreasing to nondecreasing and vice-versa during any computation is at most r. The system is 1-reversal when $r = 1$. A k-output SNP where each unbounded neuron operates with the property that it can receive spikes, but once it starts spiking it will no longer receive future spikes (but can continue spiking) would be considered a 1-reversal SNP. Moreover, as we shall see, for the results of this section, we can assume there is only one rule in each unbounded neuron, and it is $a^3(a^2)^*/a^2 \rightarrow a; 0$. Note that when $r = 0$, i.e., the system is 0-reversal, then the unbounded neurons can be deleted from the system without affecting the computation. Hence, such a system is equivalent to a k-output bounded SNP.

We need a counter machine characterization of semilinear sets. A nondeterministic multicounter machine (CM) $\mathcal{M}$ is a nondeterministic finite automaton with a finite number of counters (it has no input tape). Each counter can only hold a nonnegative integer. The machine starts in a fixed initial state with all counters zero. During the computation, each counter can be incremented by 1, decremented by 1, or tested for zero. A distinguished set of k counters (for some $k \geq 1$) is designated as the output counters. The output counters are non-decreasing (i.e., cannot be decremented). A k-tuple $(n_1, ..., n_k) \in N^k$ is generated if $\mathcal{M}$ eventually halts in an accepting state, all non-output counters zero, and the contents of the output counters are $n_1, ..., n_k$, respectively. We will refer to a CM with k output counters (the other counters are auxiliary counters) as a k-output CM.

A CM is reversal-bounded if there exists an $r \geq 0$ such that during any computation, each non-output counter is r-reversal in the sense that it alternates between nondecreasing mode and nonincreasing mode and vice-versa at most r times. Note that when $r = 0$, the counter is nondecreasing. (By definition, the output counters are 0-reversal.)

The following theorem is known [4] (see also [3]):

Theorem 3. *The following statements are equivalent for any set $Q \subseteq N^k$:*

1. *Q is a semilinear set.*
2. *Q can be generated by a reversal-bounded k-output CM.*
3. *Q can be generated by a CM with exactly k counters, all of which are output counters (hence, 0-reversal counters).*

Using the above result, we can prove:

Theorem 4. *The following statements are equivalent for any $Q \subseteq N^k$:*

1. *Q is semilinear.*
2. *Q can be generated by a reversal-bounded k-output unbounded SNP.*
3. *Q can be generated by a 1-reversal k-output unbounded SNP.*

Remark 1. We note that the theorem above holds even if we restrict the unbounded neurons in the SNP to have only the rule $a^3(a^2)^*/a^2 \rightarrow a; 0$.

4 Reversal-Bounded *k*-Train Unbounded SNPs

Now consider another restricted model called *reversal-bounded k-train SNP*. Such a system has only one output neuron. Again, there are two types of neurons: bounded and unbounded but reversal-bounded, as defined before. We say that Π generates the k-tuple $(n_1, ..., n_k)$ if, starting from the initial configuration, the output neuron O generates the spike train $aa...a$ with exactly $k+1$ outputted a's such that the interval between the i^{th} a and the $i + 1^{st}$ a is n_i, and the system eventually halts.

Theorem 5. *The following statements are equivalent for any $Q \subseteq N^k$:*

1. *Q is semilinear.*
2. *Q can be generated by a reversal-bounded k-train unbounded SNP.*
3. *Q can be generated by a 1-reversal k-train unbounded SNP.*

Remark 2. Again, we note that the theorem above holds even if we restrict the unbounded neurons in the SNP to have only the rule $a^3(a^2)^*/a^2 \rightarrow a; 0$.

5 *k*-Train Bounded SNPs

We will give a characterization of k-train SNPs all of whose neurons are bounded. A monotonic k-output CM is *sequential* if for $1 \leq r < k$, counter $r+1$ can (and must) only start incrementing when counter r has stopped incrementing after having been incremented. Hence, a sequential monotonic k-output CM has simplified rules of the forms

$l_i : (\texttt{BEGIN}(1), l_{i_1}, \cdots, l_{i_t})$
$l_i : (\texttt{BEGIN}(r+1), \texttt{END}(r), l_{i_1}, \cdots, l_{i_t})$ for $1 \leq r < k$
$l_i : (\texttt{END}(k), l_{i_1}, \cdots, l_{i_t})$
$l_i : (\texttt{DELAY}, l_{i_1}, \cdots, l_{i_t})$
$l_i : (\texttt{HALT})$

For a set $Q \subseteq N^k$, define the language (over k symbols $a_1, ..., a_k$), $L_Q = \{a_1^{n_1} ... a_k^{n_k} \mid (n_1, ..., n_k) \in Q\}$. Clearly, Q is generated by a sequential k-output CM if and only if L_Q is a regular set, i.e., accepted by a nondeterministic finite automaton (NFA).

Theorem 6. *A set $Q \subseteq N^k$ is generated by a k-train bounded SNP if and only if it is generated by a sequential monotonic k-ouput CM.*

Corollary 1. *For $k \geq 2$, k-train bounded SNPs are strictly weaker than k-output bounded SNPs.*

6 Conclusion

The results in this paper can be summarized as follow. For $k \geq 2$:

sequential monotonic k-output CMs
$=$ k-train bounded SNPs
$<$ k-output bounded SNPs
$=$ monotonic k-output CMs
$<$ reversal-bounded k-output CMs
$=$ reversal-bounded k-output unbounded SNPs
$=$ reversal-bounded k-train unbounded SNPs
$=$ semilinear sets over N^k.

In the above '$=$' means equivalent, and '$<$' means weaker.

Suppose we augment a reversal-bounded k-output unbounded SNP with one unbounded free neuron. Thus, one neuron is unbounded with no reversal bound and all the other neurons are reversal-bounded. Call such a system *reversal-bounded + 1-free k-output unbounded SNP*. This model of SNP can be simulated by a reversal-bounded k-ouput CMs augmented with one free (i.e., unrestricted counter). It is known that this model is equivalent to one with only reversal-bounded counters [4]. Hence such CMs generate only semilinear sets. Thus, we can add "= reversal-bounded k-ouput CMs + 1 free counter" at the end of the

above results. For the case $k = 1$, all the models above are equivalent, and they generate exactly the semilinear sets over N^1 (i.e., regular sets over a^*).

It follows that many closure properties (e.g., union, intersection, and complementation) hold for sets generated by the SNP models above. Similarly, many standard decision problems (e.g., membership, containment, and equivalence problems) are decidable.

References

1. W. Gerstner, W Kistler: *Spiking Neuron Models. Single Neurons, Populations, Plasticity*. Cambridge Univ. Press, 2002.
2. S. Greibach: Remarks on blind and partially blind one-way multicounter machines. *Theoretical Computer Science*, 7, 3 (1978), 311–324.
3. T. Harju, O. Ibarra, J. Karhumaki, and A. Salomaa: Some decision problems concerning semilinearity and commutation. *Journal of Computer and System Sciences*, 65 (2002), 278–294.
4. O. Ibarra: Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25 (1978), 116–133.
5. O. Ibarra, A. Păun, Gh. Păun, A. Rodriguez-Paton, P. Sosik, and S. Woodworth: Normal forms for spiking neural P systems, submitted, 2006.
6. M. Ionescu, Gh. Păun, T. Yokomori: Spiking neural P systems. *Fundamenta Informaticae*, 71, 2-3 (2006), 279–308 (also available at [13]).
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, M.J. Pérez-Jiménez, G. Rozenberg: Spike trains in spiking neural P systems. *Intern. J. Found. Computer Sci.*, to appear (also available at [13]).
12. Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg: Infinite spike trains in spiking neural P systems. Submitted, 2006.
13. The P Systems Web Page: `http://psystems.disco.unimib.it`.