Periodic Dynamical Behavior in SN P Systems and the Presence of Cycles in their Graph Representation

Gyenn Neil Ibo*
Henry N. Adorna
gvibo@up.edu.ph,ha@dcs.upd.edu.ph
Department of Computer Science (Algortihms & Complexity)
University of the Philippines Diliman
Diliman 1101 Quezon City, Philippines

ABSTRACT

In this paper, we explored how the topology of a generative Spiking Neural P system (SN P system) relates with its dynamic behavior. Specifically, we proved our claim that periodicity in an SN P system implies presence of cycles in the graph representing the system (where neurons are represented as nodes and synapses are rpresented as directed edges). Moreover, we extend the notion of periodicity from deterministic , one-rule-per-neuron generative SN P systems to the non-deterministic, multi-rules-per-neuron generative SN P systems. We also emphasize the importance of periodicity in generative SN P systems, and proved that a generative SN P system can generate an infinite number of outputs if and only if it is periodic.

KEYWORDS

Membrane Computing, Spiking Neural P Systems, Periodicity, Dynamical Aspects of P Systems, Cycles in Graphs

1 INTRODUCTION

Spiking Neural P Systems were introduced in [3] as a particular type of P Systems that abstracts and applies ideas from neurobiology. This system consists of monomembranar cells, with the synaptic connections in between them serving as media for transporting an object called a spike. Despite several restrictive characteristics of these systems, they have been found to be Turing complete [3].

Among the many peculiarities of SN P Systems, the idea of encoding information as time duration is the most distinctive. These systems are designed such that their computations rely heavily on the amount of time elapsed between particular events, and their output is likewise represented as such. This idea is derived from the fact that most of the neural impulses are almost identical, electrical signals of a given voltage [3]. Information, thus, is not encoded in the strength of these signals, but in the frequency and the time of their occurrence.

In recent years, quite a number of topics have been raised [3] and studied in this relatively young field. In [4], for instance, the already formal SN P Systems is further formalized and abstracted. The result of this particular study is an algebraic representation of the concepts in SN P Systems: the configuration of a system, the set of rules that can be applied, and the amount of spikes gained and lost by neurons are represented as vectors, the structure and relation of the rules to the neurons is represented as a matrix, and the event of a computation, or the application of the rules, is represented by multiplying a vector with a matrix. Aside from the fact that it simplifies the programmer's job of coding a program simulating SN P Systems, what is more admirable from a theoretical viewpoint is that it presents an elegant representation of a rather complex system. In [2] the foundations laid in [4] were used to explore a particular dynamical aspect of SN P Systems − periodicity. Dynamical studies and investigations on periodicity have already been conducted in other types of P Systems [1].

In this paper, we took note of the importance of the study of periodicity in SN P systems in [2], and we explored how a generative SN P system's topology relates with its dynamical behavior.

In the next section, we present the formal definition of SN P Systems. In Section 3, we recall the concepts and ideas presented in [2]. In Section 4, we proved our claim that periodicity in an SN P system implies presence of cycles in the graph representing the system (where neurons are represented as nodes, and synapses are represented as directed edges). Moreover, in Section 5, we extend the notion of periodicity from deterministic, one-rule-per-neuron generative SN P systems to the nondeterministic, multiple-rules-per-neuron generative SN P systems. We also emphasize the importance of periodicity in generative SN P systems, and proved that a generative SN P system can generate an infinite number of outputs if and only if it is periodic. We end with the final remarks in Section 6.

2 PRELIMINARIES

We present here the formal definition of SN P Systems without delay, which is a restricted variant, as adapted from [4].

Definition 1. **(SN P Systems without delay)**
An **SN P System without delay, of degree m** $m \geq 1$, *is construct of the form*

$$\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in, out),$$

where:

(1) $O = \{a\}$ *is the singleton alphabet (a is referred to as the spike);*
(2) $\sigma_1, \ldots, \sigma_m$ *are neurons of the form*

$$\sigma_i = (n_i, R_i), 1 \leq i \leq m,$$

*corresponding author: ha@dcs.upd.edu.ph.

where:
(a) $n_i \geq 0$ *is the initial number of spikes in* σ_i*;*
(b) R_i *is a finite set of rules of the following two forms:*
(i) $E/a^c \rightarrow a^p$*, where* E *is a regular expression over* $\{a\}$*, and* $c \geq 1, p \geq 1$*, with the restriction* $c \geq p$*;*
(ii) $a^s \rightarrow \lambda$ *for* $s \geq 1$*, with the restriction that for each rule* $E/a^s \rightarrow a^p$ *of type (1) from* R_i*,* $a^s \notin L(E)$*;*
(3) $syn = \{(i,j) \mid 1 \leq i,j \leq m, i \neq j\}$ *(synapses between neurons);*
(4) $in, out \in \{1,2,\ldots,m\}$ *indicate the input and output neurons, respectively.*

The rules of type (1) can be applied if the neuron σ_i contains k spikes, and $a^k \in L(E), k \geq c$. Applying this type of rule consumes c spikes from neuron σ_i and sends p spikes to all neurons to which it has an outgoing synapse.

The type (2) rules (also known as forgetting rules) can be applied if neuron σ_i contains exactly s spikes; this removes all the spikes in the neuron.

During each time unit, if a neuron σ_i can apply one of its rules, then a rule from R_i has to be applied. Note that it is possible that two or more rules in R_i can be validly applied at a particular time unit, in which case only one of them is nondeterministically chosen and applied.

This means that the rules are applied in a sequential manner in each neuron, at most one at a time, whereas several neurons can fire simultaneously, functioning in parallel.

The number of spikes in each neuron represents the configuration of the system during that time-step. For each configuration, a particular set of rules, according to the criteria described above, can be applied. This produces a sequence of configurations, which is called a computation of the system. This computation halts if it reaches a configuration where no rule can be applied. The number of steps elapsed between the first two spikes of the designated output neuron is considered as the output of the system.

What we just presented above is a formal definition of an SN P system. More often than not, however, an SN P system is presented in its graphical form. An example of a graph representation of an SN P system is shown in Figure 1. In the figure, neurons are represented as box-like figures, and synapses are represented as arrows – directed lines with an origin and destination neuron. This figure can be seen as a directed graph, where neurons are the nodes, and synapses are directed edges.

3 MATRIX REPRESENTATION OF SN P SYSTEMS

We recall here the matrix representation of SN P systems introduced in [4]. We refer the reader to the said paper for further details.

Definition 2. **(Configuration Vectors)**
Let Π *be an SN P system with* m *neurons, the vector* $C_0 = (n_1, n_2, \ldots, n_m)$ *is called the* **initial configuration vector** *of* Π*, where* n_i *is the amount of initial spikes present in neuron* s_i*,* $i = 1,2,\ldots,m$ *before the computation starts.*

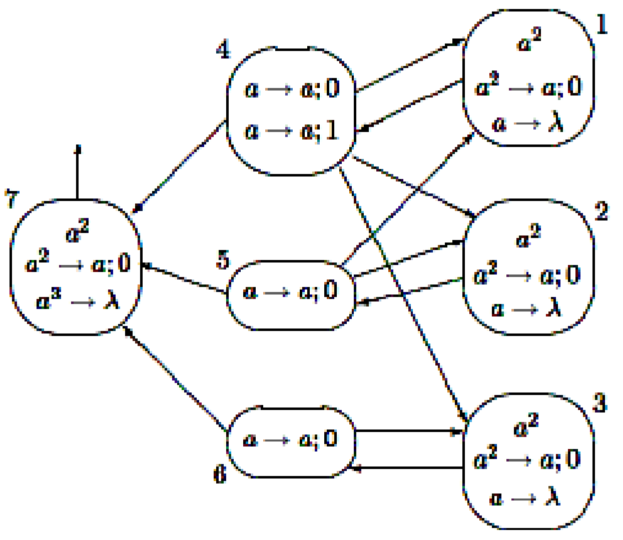

Figure 1: An SN P system generating all even natural numbers, from [3].

For any $k \in \mathrm{N}$ *, the vector* $C_k = ({n_1}^{(k)}, {n_2}^{(k)}, \ldots, {n_m}^{(k)})$ *is called the kth configuration vector of the system, where* ${n_i}^{(k)}$ *is the amount of spikes in neuron* s_i*,* $i = 1,2,\ldots,m$ *after the kth step in the computation.*

Definition 3. **(Spiking Vectors)**
Let Π *be an SN P system with* m *neurons and* n *rules. Assume a total order* d *:* $1,\ldots,n$ *is given for all the* n *rules, so the rules can be referred as* $r_1,\ldots,r_n$*, and* $E_1,\ldots,E_n$ *the corresponding regular expressions defined in Definition 2-1. A spiking vector* $s^{(k)}$ *is defined as* $s^{(k)} = ({r_1}^{(k)}, {r_2}^{(k)}, \ldots, {r_n}^{(k)})$*. The computation of* $s^{(k)}$ *is as follows:* ${r_i}^{(k)}$ *is given as 1, if the regular expression* E_i *(defined in Definition 2-1) of the rule* r_i *is satisfied and the rule* r_i *is chosen and applied, and 0 otherwise.*

The *spiking transition matrix* in the following definition will represent the amount of spikes consumed (or received) by each neuron in every application of each rule.

Definition 4. **(Spiking Transition Matrix)**
Let Π *be an SN P system with* m *neurons and* n *rules, and* d *:* $1,\ldots,n$ *be a total order for all the* n *rules. The spiking transiton matrix* M_Π *of the system* Π *is defined as follows:*

$$M_\Pi = [a_{ij}]_{nxm},$$

where a_{ij} *is equal to the following:*
= -c, if rule r_i *is in neuron* s_j *and it is applied consuming c spikes;*
= p, if rule r_i *is in neuron* s_s *(s* $\neq$*j and (s, j)* $\in$ *syn) and it is applied producing p spikes;*
= 0, rule r_i *is in neuron* s_s *(s* $\neq$*j and (s, j)* $\notin$ *syn).*

The idea of a transition net gain vector is also defined in order to algebraically represent the computations in an SN P System.

Definition 5. **(Transition Net Gain Vector)**
Let ? be an SN P system with m neurons and n rules, and C_k = ($n_1{}^{(k)}$, $n_2{}^{(k)}$,..., $n_m{}^{(k)}$) be the kth configuration vector of Π. *The transition net gain vector at step k is defined as:*

$$NG^{(k)} \;=\; C_{k+1} \text{ - } C_k \tag{1}$$

Note that the transition net gain vector NG$^{(k)}$ can also be derived as the product between the spiking vector s$^{(k)}$ and the spiking transition matrix M_Π, as stated in Lemma 3.1 in [4].. That is,

$$NG^{(k)} = s^{(k)} \cdot M_\Pi \tag{2}$$

4 PERIODICITY AND CYCLES IN DETERMINISTIC SN P SYSTEMS

In this section, we prove that periodicity in a deterministic, one-rule-per-neuron SN P system implies a presence of at least one cycle in the graph representation of the SN P system. We likewise recall the definition of periodicity in SN P systems as presented in [2]. Note also that when we say that a neuron fires or spikes, we mean that it applies one of its rules, which is a fairly common terminology used in SN P systems literature.

Definition 6. **(Computation Sequence of an SN P System)**
Let Π *be an SN P System with m neurons and m rules,* $C_0 = (n_1^{(0)}, n_2^{(0)}, \ldots, n_m^{(0)})$ *be the initial configuration vector,* $s^{(0)} = (r_1^{(0)}, r_2^{(0)}, \ldots, r_m^{(0)})$ *be the initial spiking vector, and* M_Π *be the spiking transition matrix of* Π. *The* **computation sequence of** Π *is a sequence $COMP_{seq}$ consists of configurations starting with C_0, with the proceeding elements obtained recursively by the formula,*

$$C_k = C_{(k-1)} + s^{(k-1)} \cdot M_\Pi \tag{3}$$

Definition 7. **(Periodic SN P System)**
Let Π *be an SN P System withm neurons rules, and $COMP_{seq}$ be the computation sequence of* Π. *The system* Π *is* **periodic** *if and only if there exist configurations C_k and C_p in $COMP_{seq}, k \neq p$, and $C_k = C_p$. That is, an SN P System is* **periodic (or ultimately periodic)** *if and only if a particular configuration is repeated in the sequence $COMP_{seq}$.*

Take note that Π has exactly one rule per neuron (m rules and m neurons). This makes it a deterministic system, since in every configuration there is no ambiguity of which rule to be chosen to apply in any neuron. In any deterministic system, from any given configuration C_k,, there is only possible configuration $C_{(k+1)}$ to which the system transitions. Thus, if the k^{th} element C_k in the computation sequence $COMP_{seq}$ is later repeated in the p^{th} element C_p, then the subsequence $(C_{(k+1)}, C_{(k+2)}, \ldots, C_{(p-1)})$ that immediately follows C_k and immediately precedes C_p will also be the same subsequence $(C_{(p+1)}, C_{(p+2)}, \ldots, C_{(s-1)})$, that immediately follows C_p and immediately precedes C_s, where $s = p + (p - k)$, and $C_s = C_p = C_k$. Likewise, the same subsequence will also immediately follow C_s, and so on. This forms a periodic sequence, where the period is $(p - k)$. Moreover, given that a system Π is periodic, then it implies that at least one configuration C_k during a particular time step k must be repeated at some other time-step p.

Definition 8. **(Active Neuron)**
A neuron that fires one of its rules at any point in the entire computation sequence of an SN P system is an **active neuron.**

Note that in most cases, all the neurons of an SN P system are *active neurons.* We just explicitly define this here for clarity of our discussions later on in this paper, to emphasize the fact that a neuron fires at some point of the system's computation.

Observation 1. *If a neuron fires (i.e., applies one of its firing rules), then it decreases its number of spikes.*

Firing rules are of the form $E/a^c \rightarrow a^p$, as presented in Section 2. Applying this rule means that c spikes are consumed from the neuron, thus decreasing its number of spikes. This further means that the number of spikes in a neuron can independently decrease, regardless of the events in other neurons.

Observation 2. *A neuron can not independently increase the number of spikes it contains.*

Rules in an SN P system are only either of the two forms: (1) $E/a^c \rightarrow a^p$, or (2) $E/a^c \rightarrow \lambda$, which are called firing and forgetting rules, respectively. Application of either of these rules, as stated in Observation 1, decreases its number of spikes. There is no such rule in standard SN P systems where a spike is increased within the neuron itself.

Observation 3. *An increase in the number of spikes in a neuron, say neuron i, happens if and only if another neuron, say neuron j, fires, applying a rule of the form $E/a^c \rightarrow a^p$, where $p > 0$, and a synapse from neuron j directed towards neuron i exists, that is, a synapse $(j, i) \in syn$, where syn is the set of all synapses in the system.*

Observation 4. *In a periodic SN P system, the number of spikes of each of its constituent neurons is also periodic.*

Observation 4 is evident by recalling that periodicity in an SN P system implies that the sequence of configurations in its Computation Sequence is periodic, and that the configuration is defined as a vector that is consists of the number of spikes in each neuron.

Lemma 1. *In a periodic SN P system, for all Active Neurons, there must exist another Active Neuron that has a synapse towards it. That is, for every Active Neuron n_i, there must exist another Active Neuron n_j, such that $i \neq j$, and (j, i) is an element of the set of synapses.*

Proof. A neuron, say neuron n_i, being an active neuron implies that it must fire at some point in the computation of the system. If a neuron fires, then by Observation 1, its number of spikes must decrease. Given that the SN P system is periodic, then by Observation 4, the number of spikes contained in each of its neurons must also be periodic. But since the number of spikes in the active neurons decrease whenever they apply their firing rules, at some point after firing, there must be some spikes being added to the active neuron for it to be periodic. And by Observations 2 and 3, the only way that spikes are added into a neuron is that some other neuron also fires (an active neuron), say neuron n_j, and that it has a synapse towards neuron n_i. ☐

Theorem 1. *If a finite SN P system is periodic (and at least one of its neurons is an active neuron), then there must exist at least one cycle in the graph representation of the SN P system.*

Proof. By Lemma 1, for every active neuron ni, there must exist another active neuron n_j (not necessarily unique), such that $i \neq j$, and n_j has a synapse towards n_i (i.e., $(i, j) \in syn$, the set of synapses of the system). But since every n_j is also an active neuron, for every n_j there must also exist another active neuron n_k (not necessarily unique), such that $j \neq k$, and n_k has a synapse towards n_j (i.e., $(k, j) \in syn$). This leads to a domino effect, wherein every *Active Neuron* requires an existence of another, not necessarily unique, *active neuron.* This domino effect could be consummated in a system with a finite number of neurons only if there exists at least one cyclic path of synapses between *active neurons.* ☐

5 PERIODICITY IN NONDETERMINISTIC SN P SYSTEMS

Here, we extend the notion of periodicity introduced in [2] to nondeterministic, multiple-rules-per-neuron SN P systems.

Definition 9. **(Output Configuration)**
A configuration during which the output neuron spikes. The output of this computation is computed as the time difference between this instance and the time at which the output neuron spiked for the first time. Note that there could be several Output Configurations within a single SN P system (although the output neuron is strictly only one), and the set of these output configurations is a subset of all the configurations in the system's computation tree.

Definition 10. **(Computation Graph of a Nondeterministic SN P System)**
The Computation Graph of a nondeterministic SN P system is a directed graph where the configurations are represented as nodes, and the transitions from a configuration to another is represented as directed edges. Figure 2 shows the Computation Graph of the SN P system in Figure 1.

The nodes of the graph shown in Figure 2 represent the configurations of the SN P system; inside them are the configuration vectors (described in Section 2). The directed edges represent the transitions between configurations. Configuration $(1, 1, 1, 1, 0, 0, 2)$, labeled $O.C.$ in its upper-right area, is an Output Configuration.

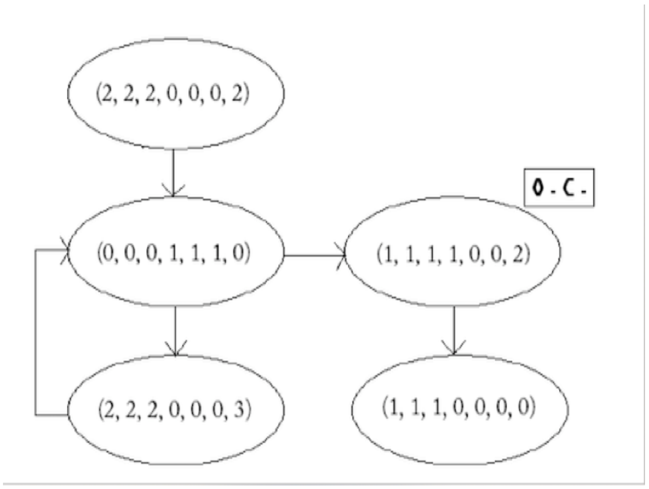

Figure 2: Computation Graph of the SN P system shown in Figure 1.

Definition 11. **(Generative Computation Path).**
A Generative Computation Path is a path in the Computation Graph of an SN P system that starts from the initial configuration and ends at the Output Configuration.

Definition 12. **(Periodic Nondeterministic Generative SN P System).**
A Periodic Nondeterministic Generative SN P system is a system whose Computation Graph includes at least one cycle.

Definition 13. **($O.C.$-Periodic Nondeterministic Generative SN P System).**
An $O.C.$- Periodic Nondeterministic Generative SN P system is a Periodic Nondeterministic Generative SN P system where there exists at least one path from any configuration node (that is part of at least one cycle in its Computation Graph) to an Output Configuration.

Theorem 2. *A finite nondeterministic generative SN P system (whose number of spikes is bounded) generates an infinite number of outputs if and only if it is an $O.C.$- Periodic Nondeterministic Generative SN P system.*

Proof. Since the system has only a finite number of neurons and bounded number of spikes, then it can only be in a finite number of possible configurations. A finite number of configurations implies a finite number of nodes in the system's Computation Graph. Moreover, every output generated by the system uniquely corresponds to a Generative Computation Path within the Computation Graph. Thus, there could be an infinite number of Generative Computation Paths within a Computation Graph with finite number of nodes if and only if there are cycles that lead to an Output Configuration. ☐

6 FINAL REMARKS

In this paper, we explored how a computing model's (specifically, SN P system) topology relates to its dynamical behavior. Since time is a crucial component in the computation process in SN P systems, then it is paramount that its dynamical behavior across time be studied. Moreover, we have proven that periodicity is an integral behavior of SN P systems for them to be able to generate an infinite number of outputs. We believe that topological connections play an important role in network-like systems such as SN P systems, Artificial Neural Networks, and the like, and thus it is the aim of our future endeavors to investigate further on more specific relations between topological properties and computing capabilities of such computing models.

ACKNOWLEDGMENTS.

The authors wish to acknowledge the support of the ERDT program of DOST-SEI, and express appreciation towards the Algorithms and Complexity Laboratory. They likewise are thankful for the encouragement, discussions and insights of their colleagues in the said laboratory.

REFERENCES

[1] F. Bernardini and V. Manca. 2003. Dynamical Aspects of P Systems. *Biosystems* Vol. 70 (2003), 82–93. Issue Issue 2.
[2] G.N. Ibo and H. Adorna. [n.d.]. Periodicity as a Dynamical Aspect of Generative Spiking Neural P Systems. In *12th International Conference on Membrane Computing.*
[3] M. Ionescu, Gh. PĂČun, and T. Yokomori. [n.d.]. Spiking Neural P Systems. *Journal Fundamenta Informaticae* Vol. 71 ([n. d.]). Issue Issue 2,3.
[4] X. Zeng, H. Adorna, M.A. Martinez-del Amor, L. Pan, and M. PĂĺrez-JimĂĺnez. [n.d.]. Matrix Representation of SN P Systems.. In *11th International Conference on Membrane Computing.*